The security and insecurity of blockchains and smart contracts

Jean-Pierre Flori,
expert in cryptography, Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)

In J.P. Dardayrol, editor of the special issue Blockchains and smart contracts: The technology of trust? of Réalités industrielles, 2017

Abstract:
The security of many blockchain applications depends not just on the construction of the chain of blocks. It also depends on the specific characteristics of the application that uses data from the chain. It further depends on the conduct of the "entities" involved in expanding the blockchain. Two flagships of blockchain technology are used to examine these various levels of security: the crytocurrency bitcoin and Ethereum Virtual Machine.

Blockchains are a key element in the cryptographic currency bitcoins and in the computing world of Ethereum, not to mention several more recent projects. The security of such applications depends not only on the construction of the underlying blockchain but also on the specific characteristics of the application built with data stored on the blockchain.[1]

The security of blockchains

To broach the question of the security of a blockchain, let us reduce the latter to its simplest form. This brings us down to an extremely simple (and even technically indigent) concept: a blockchain is a chained list where each new block points toward the preceding block. Though constantly used in the virtual computer world to store data, such a chain brings no warranty about the security of data stored in a malevolent environment. Furthermore, it is harder to share and update a distributed structure of data.

The rigidity of blockchains

If no security procedure is set up, how to be sure that a block on the chained list is not replaced with another block? To make sure, we must "rigidify" the chained list.

This rigidity is usually obtained by including in the *n*th block of data a "hash" of the *i-1* block of a fixed size (256 bits for example). To calculate this hash, a cryptographic public hashing function (which has not been parametrized by a secret, such as SHA-2 or SHA-3, among others) is applied to the block. The property expected from this function is its second-preimage resistance: to replace a block *N* in the list, it is necessary to be capable of finding another block with the same hash as the one stored in bloc *N+1*. Fortunately, the hashing functions are specially designed so that it would be an illusion to hope to accomplish such an operation.

[1] This article has been translated from French by Noal Mellott (Omaha Beach, France).

This structure can serve as a registry (ledger or register). The blocks added onto the chain are immutable. Starting from the last block added, we can be sure that none of the preceding blocks has been replaced.

Updating a blockchain

The major technological challenge of blockchains has to do with the "distribution" of the registry. It is necessary both to make sure that all parties agree on a common version of the blockchain and to keep malevolent forces from taking control over the adding of blocks onto the chain. This "Byzantine generals problem" had already received attention from computer scientists in relation to models of "permission" where all players are known and controlled.

However the many existing protocols for reaching a consensus in a controlled model where all players are known no longer work when unauthenticated forces take part in the protocol, or withdraw from it, as they want. Bitcoin's principal innovation is to have proposed a protocol for reaching a consensus in its "permissionless" environment by using cryptographic "proof-of-work" puzzles.

Proof of work

The principle underlying a proof-of-work system is the following: to be able to propose a new block for validation so as to be added onto the blockchain, a difficult problem has to be solved.

In the Bitcoin network, this proof of work resorts, again, to cryptographic hashing functions. Given the preceding block's hash $H(B_{i-1})$, and given the data to be incorporated in the new block B_i (a list Li of transactions to be validated), the problem is, in the main, to find a prefix N_i such that the new block — $B_i=(N_i, H(B_{i-1}), Li)$ formed by the concatenation of these elements — satisfies the condition that the hash $H(B_i)$ starts with i null bits. This problem (more difficult than the one mentioned previously) comes close to the problem of calculating the first-preimage resistance of a hashing function.

For all modern cryptographic hashing functions, no better method yet exists for solving this problem than to propose random N_i values and calculate the corresponding hash (it is possible to omit a few runs of the hashing function). It suffices to set the value of i so that a new block is generated about every ten minutes (at present approximately 70 minutes for Bitcoin).

A proof-of-work system does not, by itself, lead to an agreement when two valid blocks are transmitted over the network at the same time. This is called a "fork". When this happens, Bitcoin makes a start at reaching a consensus by "doing nothing". Each node in the network chooses randomly one of the two valid blocks for eventually adding it onto its local copy of the blockchain; it works on that block. Over time, it is highly probable that the one of the two choices will lead to a longer blockchain than the other. The consensus is that all nodes will then switch to using the longest blockchain.

An important security parameter is to set the difficulty of the cryptographic puzzle to be solved. If a malevolent player were to control more than half the network's computational power, he would have a probability of more than 50% of being able to generate a block before the other nodes. It is plausible (but not formally proven!) that Bitcoin's strategy will prevent players from rewriting the chronology by creating their own fork from the node of their choice and adding new blocks on it until it becomes longer than the official blockchain. This strategy fends off cases of double spending, when a user transfers bitcoins, waits for enough blocks to be added to the chain so that the recipient thinks the transaction has been validated, and then rewrites the chronology so as to replace the initial transfer with another. Furthermore, this strategy prevents censorship, in the case of an attacker who makes a longer fork when he comes upon a transaction on the chain that he does not want.

Formal security

An axis of current research is to formalize the protocol by producing formal proof of its security. Among the concepts involved are:

- spatial coherence: various honest players must share the same vision of the official blockchain (or at least a common prefix, so as to take account of a slight desynchronization or of brief forks).
- temporal coherence: over time, an honest player's blockchain has to maintain its fixed prefix (in other words, his vision of the blockchain can change but marginally, at the end of chain).
- growth: the "official" chain's length increases for as long as as the number of players contributing to it is sufficient.
- quality: a satisfying proportion of the blocks generated by honest players end up being incorporated in the chain.

All these studies place more or less restrictive conditions on building models of the network connecting players and on the population dynamics of the players. However they propose improvements over the original Bitcoin protocol for increasing the average speed for validating blocks (via secondary chains and mixed protocols of permissioned and permissionless consensus).

Some alternative cryptocurrencies with less mining power than Bitcoin's or Ethereum's have been attacked by groups who were able to rewrite the blockchain and spend several times their assets by pooling a computational power of more than than 51%. This happened, for example, to the Ethereum clones Krypton and Shift.

Security on the Bitcoin network

In addition to a blockchain, Bitcoin proposes a cryptocurrency. Each player has a pair of keys (the one private, the other public) for a cryptographic signature; and each transaction is recorded using a very simple script language mainly for the instruction "transfer the bitcoins of the transaction *y* toward the address *s*", along with a signature in which: *a)* the address *z* is the hash of a public signature key; and *b)* the transaction's number points toward a previous transaction where bitcoins were transferred to the public key corresponding to the private key that generated the signature *s*.

To spend the bitcoins transferred to the address *z*, it is necessary to produce a cryptographic signature using the associated private key. Only the owner of the private key can do that if the underlying cryptographic signature procedure is secure and, of course, if the private key has not been disclosed or stolen.

Several procedures are adequate for cryptographic signatures. Most of them rely on mathematical problems such as the factoring of big whole numbers or the resolution of a discrete logarithm over chosen groups. The cryptographic signature used by Bitcoin relies on this second type of problem, specifically on the difficulty of solving a discrete logarithm over an elliptical curve (the elliptic curve digital signature algorithm, ECDSA).

Mining pools

The proof-of-work system has a deviant effect. Besides the high energy cost of mining, most mining activities are concentrated in areas, close to the plants that produce application-specific integrated circuits (ASIC), where electricity is cheap.

Persons who want to mine bitcoins but lack the computational power for doing so (profitably in relation to the investment in energy) group together to form mining "pools". A server centralizes the calculations made by all miners in the pool, and redistributes winnings in proportion. The person in charge of this server thus has full control over the pool's computing power. The possibility cannot be excluded that a single player thus ends up with more than half the network's computational power — thus undermining the blockchain's foundational principles.

Various proposals have been made to deal with these problems, such as proof-of-stake or proof-of-space/capacity.

"Light customers"

Another abuse of bitcoins casts doubt on the "distributed" aspect of the registry that provides the grounds for confidence in this cryptocurrency. To make sure that a transaction has been validated, each player would have to store the whole blockchain to check the validity of the transactions received. Since he Bitcoin blockchain now amounts to several gigabytes of data, downloading it takes time. For this reason, so-called "light customers" do not store the blockchain but turn toward the rest of the network to know for sure whether a transaction has been made.

Electronic wallets

Several users choose to delegate the management of their electronic wallets (and thus of their private keys) to specialized websites. The latter become the target of choice of hackers who try to steal or destroy the cryptocurrency. Several attacks of this sort have been made against sites for storing or exchanging bitcoins (*e.g.*, MtGox). There is no technical solution since the "trusted third party" is no longer able to restore customers' secret keys.

Security on the Ethereum network

Ethereum uses a blockchain like Bitcoin: users' are grouped in blocks by miners who verify whether the transactions are valid and try to solve a cryptographic puzzle in order to be authorized to add the new block to the chain. Whereas Bitcoin uses a simple script language for monetary transfers, Ethereum allows for using a script language with Turing-completeness, thus deploying a "virtual computer world". Each "transaction" can create or execute a program in a distributed way by all miners. Unfortunately, this potential introduces new security problems no longer of a cryptographic sort but related to computer programming and programming languages.

Ethereum Virtual Machine and Solidity

The data stored on the Ethereum blockchain serve to determine the current state of the fully distributed Ethereum Virtual Machine. A “balance variable”, associated with each user and each program stored on the blockchain, represents the quantity of the cryptocurrency “owned”.

To make it easier to design programs for deployment on the Ethereum Virtual Machine (EVM), a high-level language called Solidity is used prior to compilation. The semantics for calling a Solidity function are rather complicated. In particular, each program can define a callback function.

DAO attacks

The principle of the DAO (decentralized autonomous organization) program used to be relatively simple: receive from users donations for other users and then transfer these donations to the end user upon request (a sort of virtual treasure). The programming code can be simplified down to two functions:

- *donation (address)*: This function links the amount of cryptocurrency when the function is called to an address in a table inside the DAO program.
- *withdrawal (amount*): This function allows a user to ask the DAO program to transfer to his address a given amount of the cryptocurrency.

Unfortunately, the syntax used by the DAO program had the property of calling the callback function of the calling program. A malevolent user could thus write a program requesting the withdrawal of a legitimate amount of the currency by calling the withdrawal function, and its callback function made a similar call, thus making it possible to retrieve twice the amount due because of a bug in the original program.

Conclusion

The security problems of blockchains and their applications are numerous. A flaw at any level of the chain can have catastrophic consequences. Many innovations have to be made before this technology reaches a level of maturity and confidence that will justify using it for critical applications.