Algorand's Approach to the Right to Be Forgotten

By Silvio Micali

1. Introduction

Article 17 of the European Union's General Data Protection Regulation establishes the right to erasure of personal data ('right to be forgotten', RTBF). To be sure, data privacy laws and different forms of the RTBF are not unique to the European Union. Other legal systems, such as Brazil, Australia, or Canada have similar or stricter rules. But such laws exceed the scope of this blog, which is solely concerned with the European Union's RTBF.

The RTBF applies to personal data only: name, birthdate, government identifier, residential address, employer, education, bank account #, credit card #, blood type, gender, sexual orientation, marital status, language, disability, religion, etc. More generally, it may apply to any piece of data which, when considered alone or in combination with other data in the possession of a data collector, can identify the data subject.

This said, the RTBF does not give the data subject an absolute right to erase her data whenever she wants. For example, when applying for a mortgage, the borrower may consensually give the lender all kinds of relevant personal information to keep at least for the duration of the loan, in which case the borrower has no RTBF. More generally, the RTBF does not apply to non-personal data. It does not guarantee privacy, for instance, about the price paid to purchase a given piece of real estate. Such information may be protected contractually, but not under the RTBF or other data privacy laws.

My Focus

In strict data privacy jurisdictions, the basic rule on the storage of personal data requires that such data must be (1) stored or processed with the specific consent of the data subject and (2) tagged and clearly associated with such consent. How this consent and tagging requirements are implemented is a very important and challenging topic, but it is not my focus here. Neither is it my goal to explain in which cases the RTBF will be upheld in any given jurisdiction.

My focus is on personal data that, *after* having been made available (for whatever reason), must be erased (no matter how, why, and by whom this erasure is requested).

More precisely, my focus is on RTBF compliance for truly decentralized, permissionless blockchains in general, and for the Algorand blockchain in particular.

The RTBF and the Blockchain?

The concept of an RTBF-compliant blockchain may appear nonsensical: RTBF and blockchains do not seem to mix, like oil and water. Blockchains are about transparency and immutability, while the RTBF is about erasing personal information that was previously public, if the proper conditions are satisfied.

This conceptual difficulty is exacerbated when the blockchain is truly decentralized and permissionless. We should appreciate that such chains are fundamentally different from, say, the Internet and search engines. Google controls multiple servers all other the world and maintains its own centralized database. Millions and millions of users may request Google to download specific pages of this database, but they do not participate in its maintenance and growth. It is thus easy for Google to handle an RTBF request to de-link a given page. If the request is legitimate, then it will de-link that page. And if it continues to make it accessible, the individual who had the right to demand its removal may sue Google in court. Case closed. Simple and familiar.

By contrast, a distributed blockchain is maintained by a multiplicity of users all over the world. So, *who is responsible for properly deleting personal information?* Or, in crasser terms, *whom does one sue?*

Moreover, in a blockchain, all information is cryptographically bound together, so that one cannot naively remove any information, personal or not, without invalidating the entire blockchain. So, *how can any given piece of information be securely removed?*

More generally and more fundamentally, for a truly distributed and permissionless blockchain, *what should RTBF compliance mean?*

Resisting Temptations

To be clear, we should not just seek a blockchain that complies with all *current* RTBF regulations. No matter how appealing this may sound in the short term, this `solution' would be quite *ad hoc*. The RTBF is not written in stone. As for all things legal, and for all things human as well, it will continually evolve, and a blockchain should continue to be RTBF-compliant as regulations change over time. We need an approach as general as possible.

Moreover, we should not resign ourselves to putting a few trusted entities in charge of the chain or of the RTBF. Centralization is a common temptation but runs counter to the very spirit of a blockchain. True: a few trusted parties could easily re-write the chain and expunge any personal data that must be forgotten. But they could also re-write history, in any way they want, the moment they no longer remain trustworthy. Smart money bets that this moment would arrive soon enough…

My Take

True decentralization is a great source of security. Transparency is a major source of trust. And the right to be forgotten is a fundamental human right. We must thus do our best to square the circle, delivering a truly decentralized and transparent blockchain capable of implementing RTBF regulations. Those of today and those of the future.

2. A General Problem for Blockchains

Two Types of Transactions.

A blockchain can safely store all kinds of transactions. For simplicity, I will consider just two:

- *Data transactions*. Such a transaction T on a blockchain makes some specific data, D, available to everyone and trusted to be inalterable. A data transaction T may also include some personal information, I. If so, we write $T = (D, I)$.

- *Payments*. In a blockchain, a payment P includes the payer's public key, the payee's public key, the amount of money transferred from the first key to the second, and the payer's digital signature authorizing the transaction. We refer to all this information as the monetary transfer proper, M. But a payment P may include other information. In particular, some personal information, I. If so, we write $P = (M, I)$.

Of course, these two types of transactions may overlap,[1] but assuming that they do not will simplify our discussion. Also, payments are just a special type of transfers of general assets, and whatever we say about payments (and balances) apply to these general transfers as well.

To be sure, monetary transfers are hardly personal information and are not directly affected by the RTBF. However, the RTBF affects a payment $P = (M, I)$, if the personal information I needs to be forgotten. For instance, in a blockchain loan, the borrower may be obliged to make a series of monthly payments to the lender, and in each of those she may want (or be obliged) to include information that identifies her. Thus, in some jurisdictions, she may have the right to have this personal information erased after her loan has been paid off.

Similarly, the RTBF may not apply to a specific piece of data D, but may affect a data transaction $T = (D, I)$. As it turns out, handling RTBF requests so as to preserve basic blockchain functionalities is harder for data transactions than for payments.

[1] For instance, a payment P may also include some separate data D that is not personal information, nor a monetary transfer, that must be posted permanently on the blockchain: $P = (M, I, D)$.

The Problem with Legacy Blockchains

Legacy blockchains, such as Bitcoin, are based on the unspent-transaction-output model and require complete knowledge of all the past blocks to validate new payments and blocks. Let PK be a public key (of, say, the Bitcoin protocol) that receives an amount of money m (from another public key) in a payment P of a block B. If, in a subsequent payment P^* of a later block B^*, PK transfers all or part of m (to yet another public key), then P^* includes a pointer p to the original payment P. To validate such a new payment P^*, one must (1) `follow' the pointer p to look up the payment P in block B, (2) verify that PK did indeed receive an amount of money m, and (3) inspect all blocks between B and B^* so as to verify that PK did not already spend m.

Having to consult past transactions in order to participate in the consensus protocol (i.e., in order to validate new transactions and generate new blocks) makes it *technically impossible* for legacy blockchains to satisfy RTBF requests about personal information included in payments.

Here is why. Let $P = (M, I)$ be one such payment and let it be cryptographically secured in a past block B. If, in response to an RTBF request, the personal information I were erased, then P would immediately cease to be cryptographically validated within B and so would the monetary transfer M. In a sense, no one could prove that M really happened. So, the specific amount of money, m, that M transfers to the payee would `vaporize.' If the payee had not already spent it when the information I got erased, she would no longer be able to spend it. Should she try to do so after the erasure of I, she would need to provide a pointer to P. But anyone following such a pointer would find no proof that she ever received the amount of money m.

Let us now consider the case of a data transaction $T = (D, I)$. First of all, note that, though T itself is not a payment, a money-vaporizing problem similar to that described above continues to arise if T contains a `posting fee.' Indeed, such a fee is a form of money transfer (*e.g.,* to the miner who included T in a new block that he successfully appended to the chain). Therefore, within a legacy blockchain, fulfilling an RTBF request about T would cause the retroactive erasure of money transfers, both explicit and implicit — certainly not a desirable outcome!

Assume now that $T = (D, I)$ does not include any transaction fee and that it is cryptographically secured within a block B. That is, D and I are cryptographically secured in B *together* rather than individually. Thus, similarly to the payment case discussed above, D also vaporizes the moment I gets erased in response to an RTBF request. Once again, not a desirable outcome. Presumably, in fact, the information D was posted on the blockchain to ensure its continual availability and to enable subsequent transactions to rely on it. Indeed, as for the case of payments, *posted data affects subsequently posted data*.

Balance-Based Blockchains

A newer class of blockchains, which includes Ethereum and Algorand, handles payments differently from legacy blockchains. In these newer chains, in order to validate new payments, the participants in the consensus protocol are not required to look up and validate past payments. Rather, they need only keep and update a small amount of information: namely, the current balance of each key in the system.[2] A blockchain that so operates is a *balance-based blockchain* (BBB).

Different BBBs can have different consensus protocols. For instance, that of Ethereum is currently based on proof of work, while that of Algorand is based on *pure proof of stake*. But whatever their consensus protocol, Algorand, Ethereum, and all other BBBs can validate new payments and be RTBF-compliant. Let us see how.

In any BBB, let B be the latest block, u a participant in the consensus protocol, and PK a public key. Then, *by definition*, u knows the current balance of PK, bal_{PK}. That is, she knows that the amount of money available to PK is bal_{PK}, after all payments in the chain, up to and including those in block B, have been executed. (In other BBBs, u may know bal_{PK} by continually monitoring the chain form the start, or by receiving bal_{PK} form a source she trusts. In Algorand, u has a provable way to learn the correct value of bal_{PK} at any block.)

Assume now that an RTBF request is issued to erase the personal information I of a payment $P = (M, I)$ in a prior block A. Then u herself can go ahead and delete I from any copy of the BBB she may have. Such erasure may prevent her (or anyone else who does the same) from proving to others that the monetary transfer M really occurred in block A. Nonetheless, this proving inability does not change the amount of money currently available to PK. This is the case, whether or not PK was the payer or the payee in P. Thus, u knows that, after the personal information I of P has been deleted, the current balance of PK continues to be bal_{PK}.

Accordingly, to check whether all payments made by PK in a newly proposed block C are valid, u need only check that the sum of all amounts of money that PK transfers via its payments in C does not exceed bal_{PK}. If C is added to the chain, then u updates the balance available to PK and that of any other public key making or receiving payments in C. The same is true for the current balance of any public key in the system.

In sum, BBBs can successfully handle RTBF requests about payments.

[2] Note that, at every block, the balance of a given key comprises not only the amount of the native currency available to it, but also the stable coins and all other fungible and non-fungible assets that the key owns.

The Problem with Balance-Based Blockchains

BBBs can successfully comply with RTBF requests about payments for the following simple reasons: (a) balances capture the essential information necessary to process future payments, (b) balances do not contain personal information and are unaffected by RTBF requests, and thus (c) balances could be correctly kept even if RTBF requests demanded the erasure of all past payments.

BBBs, however, cannot successfully comply with RTBF requests about data transactions, because data transactions can be interdependent too and because there is nothing equivalent to a balance for general data. That is, it is hard to distill in a compact piece of information what is essential in an entire sequence of general data transactions. Thus, if the personal information I in a data transaction $T = (D, I)$ were to be erased in response to an RTBF request, the data D would automatically cease to be authenticated, with potentially disastrous consequences for future data transactions that should have depended on D.

3. The Algorand Solution

One person's right ends when another's right begins. The RTBF is a crucial right. But the trust generated by a truly distributed blockchain is also a matter of great public interest. It would be a pity to give up the latter when safeguarding the former. *Can we have them both?* Yes indeed.

A decentralized blockchain works thanks to the efforts of two categories of users:

1. *The consensus participants*, who validate new transactions and generate new blocks, and
2. *The information service providers*, who enable ordinary users to access information stored in already generated blocks.

(Ordinary users may just transact, store themselves already generated blocks, if they so want, or query information service providers about data stored in the chain, when they need them.)
Algorand enables consensus participants and information service providers to comply with the RTBF without any drawbacks, for themselves, the blockchain, or the ecosystem at large.

The Main Idea in a Nutshell

Algorand separates erasable from non-erasable data and guarantees the post-erasure integrity of a block by separately storing (and never erasing!) the hash of any erasable data.

To this end, Algorand modifies the traditional structure of blocks and transactions.

The Traditional Block Structure

In a chain, the ith block, B_i, has two components: (1) *the block's data,* BD_i, which contains the sequence of block's transactions, $T_1, \ldots, T_n$, as well as the signatures of the users who issued them, and (2) *the block's header,* BH_i, which cryptographically secures the block's transactions:

$$B_i = (BD_i, BH_i),$$

In its simplest form, BH_i comprises

- the hash of the previous block's header;
- the hash h_j of each transaction T_j, $h_j = H(T_j)$; and
- additional data (e.g., the block number i, time information, and so on).

In symbols, $BH_i = (H(BH_{i-1}), h_1, \ldots, h_n, AD)$.[3]

The cryptographic hash function H essentially guarantees that one cannot even minimally change a quantity Q without also causing a change in the hash value $H(Q)$. Accordingly, given the block header BH_i, to check that a transaction T_j belongs to the corresponding block and has not been altered, one hashes T_j so as to produce the result $H(T_j)$ and then checks whether this hash value indeed coincides with the value h_j that is part of BH_i. Notice that `chaining' the block headers (i.e., including in a header the hash of the previous header) ensures that no one can undetectably alter any header.

This security is a double-edged sword with respect to the RTBF. If T_j has a personal information component, $T_j = (X_j, I_j)$, then the corresponding hash value in BH_i is $h_j = H(X_j, I_j)$. Assume now that, in response to an RTBF request, I_j must be removed from the blockchain. Then, a consensus participant or an information service provider may easily comply with the request by erasing I_j and substituting (X_j, I_j) with just X_j. However, after forgetting I_j, one will no longer be able to prove that X_j belongs to the blockchain. Indeed, being X_j different from (X_j, I_j), the hash value $H(X_j)$ will differ from the hash value $h_j = H(X_j, I_j)$ that is part of BH_i. So, after expunging I_j, it becomes impossible to verify the authenticity of X_j.

To implement the discussed RTBF-compliant strategy, Algorand must avoid this problem.

[3] Hashing each transaction T_j individually is conceptually simple and enables one to verify that T_j has not been altered without relying on or disclosing any other transaction. However, it requires that a block's header includes one hash value for each of its transactions. It is more efficient for a block's header to include a single collective hash: the Merkle hash of all transactions together. This special hash still allows one to verify that each transaction in the block has not been altered without involving any other transaction. This is indeed the way Bitcoin hashes its block transactions. Whatever we say about individually hashed transactions continues to be true for Merkle hashed transactions. But assuming that each transaction is individually hashed spares the reader the details of Merkle hashing.

Algorand's Transaction Structure

Essentially, Algorand introduces two new, related but separate, transaction fields: an *erasable* field and a *random* field. The first enables one to store information, E, that might be subject to erasure at a later time; and the second a random string, R, that helps the removal of E, if needed at a later time, without disrupting the rest of the information, X, the transaction may contain. The erasable (respectively, the random) field of a transaction is allowed to be empty, in which case $E = 0$ (respectively, $R = 0$). Accordingly, a transaction T can always be written as

$$T = (X, E, R).$$

As we shall argue, X continues to remain securely stored in the blockchain, independent of any possible RTBF requests. We refer to X as the *essential information* of transaction T.

We call a transaction $T = (X, E, R)$ *RTBF-honest* if all personal information that might possibly be forgotten solely appears in E, and *RTBF-dishonest* otherwise. Honest users solely issue RTBF-honest transactions.

Algorand's Block Structure

For an Algorand block $B_i = (BD_i, BH_i)$,

- BD_i continues to include the sequence of the block's transactions,

  $$T_1 = (X_1, E_1, R_1), \ldots, T_n = (X_n, E_n, R_n)$$

  together with the signatures of their issuers. The only change in the transaction format.

- BH_i includes, for each transaction $T_j = (X_j, E_j, R_j)$ in the block, two hash values:

  $$h_j' = H(E_j, R_j) \text{ and } h_j = H(X_j, h_j').$$

  That is, $BH_i = (H(BH_{i-1}), (h_1', h_1), \ldots, (h_n', h_n), AD)$.

Algorand's Response to RTBF Requests

Assume that an RTBF request is made about an RTBF-honest transaction $T_j = (X_j, E_j, R_j)$ in a prior block B_i. Then, in response to such a request, a consensus participant or an information service provider will substitute (X_j, E_j, R_j) with X_j in her own copy of BD_i. That is, she will

- Erase E_j and R_j from the block's data BD_i;
- Continue to store X_j in BD_i; and
- Continue to keep (h'_j, h_j) in the block's header HB_i.

Assume now that an RTBF request is made about an RTBF-dishonest transaction T_j. Then, in response to such a request, a consensus participant or an information service provider will delete the entire transaction T_j in her own copy of BD_i, but continues to keep (h'_j, h_j) in the block's header HB_i.

(In either case, upon learning about an RTBF request about T_j, an honest ordinary user, who happens to store block B_i herself, may act in the same way.)

Discussion

Note that Algorand's block headers are not affected by $RTBF$ requests. Indeed, new block headers continue to be generated, as the chain grows, but remain unaltered, and are in fact unalterable, once generated.

As long as a transaction $T_j = (X_j, E_j, R_j)$ in a block B_i is not subject to an RTBF request, the unalterability of the entire T_j, including that of E_j and R_j continues to be guaranteed by the blockchain. In fact, given (X_j, E_j, R_j), one can first compute the hash value $v = H(E_j, R_j)$, then the hash value $H(X_j, v)$, and finally verify that they coincide with the two hash values h'_j and h_j securely stored in the block's header BH_i.

If an RTBF request is made about an RTBF-honest transaction $T_j = (X_j, E_j, R_j)$ in a block B_i, then only the authenticity of the essential information X_j continues to be guaranteed by the blockchain. In fact, as soon as the request is made, E_j and R_j are removed from the block's data BD_i, but not X_j. Nor is the pair of hash values (h'_j, h_j) removed from the block's header BH_i. Thus, to verify the authenticity of X_j, one retrieves the pair (h'_j, h_j) from BH_i, computes the hash value $H(X_j, h'_j)$, and verifies that the so obtained result indeed coincides with the value h_j.

If an RTBF request is made about an RTBF-dishonest transaction $T_j = (X_j, E_j, R_j)$ in a block B_i, then , as already said, the entire T_j is removed from the block's data BD_i. If T_j is a payment, then its removal does not affect other payments in the chain, because Algorand is balance-based. But if T_j were a data transaction, the meaningfulness of subsequent data transactions that rely on T_j's essential information X_j could be compromised. Thus, if the issuer of T_j is not malicious, *it is in her very interest to ensure that T_j is an RTBF-honest transaction.*

In an RTBF-honest transaction $T = (X, E, R)$, the randomness of the value R guarantees an higher level of compliance with the RTBF. To see this, assume that no random string R were used. That is, assume that all transactions T and block headers BH_i were of the form

$$T = (X, E) \text{ and } BH_i = (H(BH_{i-1}), \dots, (v', v), \dots, AD),$$

where $v' = H(E)$ and $v = (X, v')$. Assume now that E directly identified a famous person: for example, let $E =$ *"Bill Gates"*. Then, in response to an RTBF request, the blockchain might very well erase E, but the blockchain still enables any user suspecting that the transaction involves Bill Gates to *prove* that this is the case. Indeed, such a user can compute $H(Bill\ Gates)$ and check that the so computed value coincides with the securely stored value v'. Bill would have all the right to be upset… By contrast, when $h' = H("Bill\ Gates", R)$, provided that R was randomly selected, once "Bill Gates" and R have been erased, no one knowing h' and suspecting that the transaction T refers to Bill Gates could confirm to herself or prove to others that this is the case.

In an RTBF-dishonest transaction $T = (X, E, R)$, however, R may not be random at all. Rather, the issuer of T may choose $R = 0$ to enable anyone guessing E to easily confirm the correctness of her guess. This subtle difficulty is avoided by `forcing' T's issuer to choose R in a random manner. We accomplish such forcing by slightly adapting the verifiable-random-function techniques pioneered by the Algorand blockchain. This approach is, however, too technical for this blog. A simpler but reasonable alternative is given in this footnote.[4]

---

[4] When a transaction $T = (X, E, R)$ is deemed eligible to be stored in a new block i, the block proposer randomly and independently chooses a string R', which she includes in BD_i together with (X, E, R), and the two hash values that she includes in BH_i are $h' = H(E, R, R')$ and $h = H(X, h')$. It is easy to see how, with these simple modifications, one can still verify the authenticity of the entire T, before an RTBF request is made, or that of just X, after an RTBF request has been made. It is also easy to see that, as long as one of R and R' is random, when E is erased it is impossible for anyone to correctly guess E and confirm the correctness of the guess.

It is also possible to have the block proposer, rather than the T's issuer, choose R in a forcible and rigorous way. This solution is actually very efficient, but too technical for the purposes of this blog.

Algorand's RTBF-Compliant Information Service Providers

Information service providers perform two crucial functions:

(a) Enabling new users who wish to join the consensus protocol to obtain the information they need (*e.g.,* in a BBB, the balances at the latest block); and
(b) Enabling users who cannot or do not store the entire blockchain to access the specific pieces of information they need (*e.g.,* a given block or a given transaction).

To be sure, handling these functions both *securely* and *efficiently* is a problem.[5] Algorand, however, has developed a unique technology to solve this problem. Essentially, Algorand's technology enables an information service provider to answer each user query with a *short and easy-to-verify proof* that the answer is indeed correct. Importantly, such a proof is solely based on the genesis block, the only block that can be considered unambiguously known.

I shall devote a separate blog to this new and exciting technology of Algorand. Here, I wish only to emphasize that, when queried about a transaction $T = (X, E, R)$, an honest provider returns (X, E, R), with a proper proof, if no RTBF request has been made about T. Else, it returns just X, again with a proper proof, together with proper evidence that T was subject to a legitimate RTBF request.

The proper authorities can easily check whether an information service provider is so RTBF-compliant. For instance, keeping *incognito*, they can query the provider and check whether it returns information that had to be forgotten. In principle, they could also check whether the provider still stores any information that had to be forgotten, but this would be more complex. An analogous complexity arises for search engines. Assume that Google (1) is asked to de-link a page p containing personal information, (2) promises to do so, but (3) actually fails to comply with the request. Then, it is easy for a privacy authority to find out that Google is still disclosing p. It is much harder, however, for the authority to check whether Google is *no longer in possession of page* p. In other words, it is hard to prove a negative.

Note that whether an ordinary user continues to keep personal information that had be forgotten is another matter. Such keeping is analogous, in the case of search engines, to that of an individual user who continues to keep a page p that was previously de-linked. It is well established that the RTBF does not require the erasure of *all* the copies of to-be-forgotten information *ever* made and stored by *anyone*. In September 2019, the Court of Justice of the European Union held that Google would be

[5] Note that, for either function, the amount of information a user needs to obtain may be quite small, even if the blockchain has already grown to be very large. To obtain it securely, however, users of other blockchains need to download the entire chain, which is definitively not efficient. Else, they need to trust someone to give them the correct information, which is not secure. Sooner or later, some user will rely on incorrect information, with possibly disastrous consequences.

RTBF-compliant in so far as searches launched on Google engines within the EU (*e.g.,* google.fr, google.it, etc.) would not yield the erased or de-linked information. The Court did *not* demand that Google erase all copies of p in cyberspace.

Finally, it should be appreciated that, with Algorand, honest information service providers do not merely abstain from providing information that had to be forgotten. They actually erase any copy of this information they themselves possess, and yet remain capable of continuing to work correctly.

4. Final Remarks

Ease of Use

To harvest the power of this technology, we need a convenient, secure, and indeed blockchain-based way to communicate to consensus participants and information service providers which personal information should be forgotten, and which has (consensually) been exempted from the RTBF. For instance, to certify that a user u has indefinitely (respectively, until a given time t) waived her RTBF about her personal information E in a transaction $T = (X, E, R)$, society may agree that it suffices for u to co-sign digitally T (respectively, (T, t)).

By the way, Algorand's *atomic transaction* technology easily and speedily guarantees, indeed at layer 1, that the issuer of T and u can independently sign T, without worrying about who signs first, with the guarantee that T will be posted on the Algorand blockchain only if it is digitally signed by both parties.

You may read more about the advantages of Algorand's atomic transactions and other layer-1 smart contracts in my previous blogs.

Forward Compatibility

Whatever its consensus protocol, any blockchain can become RTBF compliant simply by switching to Algorand's transaction and block structure. Should RTBF rules become stricter at a later time, RTBF compliance will be assured, going forward, by including the newly protected personal information in the erasable component of all subsequent transactions.

Partial Adoption

Note that Algorand's RTBF approach still works when any (or even all) of the consensus participants do not erase the information that must be forgotten: indeed, so long as Algorand's transaction and block structures are in place, honest service information providers can still work in an RTBF-compliant way. And dishonest ones can be identified and sued in court.

General Commitments

The essence of Algorand's RTBF approach is to separate, in a transaction T, personal information I from any associated information X, so as to be able to erase I while maintaining the availability and the inalterability of X. Specifically, I is first probabilistically hashed (i.e., hashed together with some randomness that is initially stored), and then X is hashed together with the first hash value.

It should be realized, however, that (probabilistic) hashing is a special form of *commitment*. Indeed, many and sophisticated commitment schemes may be alternatively used here. Such a scheme enables one to (1) `pin-down' a chosen value, while keeping it secret for a while, and then, when deemed appropriate (2) reveal the value in a provable manner, that is, by guaranteeing that the revealed value is indeed the originally chosen one.

Essentially, given a value x, a committer computes another string $C(x)$, the *commitment* (to x), and another value d, the *de-commitment*. Traditionally, the committer publicizes $C(x)$ and keeps secret d. The commitment $C(x)$ satisfies two properties. First, it prevents anyone, even the committer herself, to be able to modify the value x at revealing time. (The *binding* property.) Second, it prevents anyone else from obtaining any information about x, before it is revealed. (The *secrecy* or *hiding* property.) To reveal the committed value x, the committer also reveals d, so as to enable any one to verify that x is indeed the original string pinned down by the commitment $C(x)$. In the simple implementation of our strategy described earlier in this blog, the committed value is the personal information E; the commitment is $C(E) = H(E, R)$; and the decommitment is $d = E, R$.

Notice that any commitment scheme can be used within our strategy. Also notice, that our strategy makes a non-traditional use of a commitment scheme. In fact, in a typical commitment application, the decommitment information is kept secret, because one wants to hide the committed value until it is revealed. In our strategy, instead, both the committed value and the decommitment information are initially made available. However, once the committed value and the decommitment information are both deleted, the secrecy property of the commitment scheme guarantees that E becomes and remains secret.

5. In Sum

Algorand *embraces the RTBF by design*. It enables and in fact encourages users and developers alike to work with the RTBF. Fundamental rights benefit us all, but scientists and legislators must work together to ensure that, whenever possible, the enforcement of these rights remain compatible with other great benefits, such as the trust and transparency provided by a public and truly decentralized blockchain.

SILVIO MICALI | Founder, Algorand

Silvio Micali has been on the faculty at MIT, Electrical Engineering and Computer Science Department, since 1983. Silvio's research interests are cryptography, zero knowledge, pseudorandom generation, secure protocols, and mechanism design and blockchain. In particular, Silvio is the co-inventor of probabilistic encryption, Zero-Knowledge Proofs, Verifiable Random Functions and many of the protocols that are the foundations of modern cryptography.

In 2017, Silvio founded Algorand, a fully decentralized, secure, and scalable blockchain which provides a common platform for building products and services for a borderless economy. At Algorand, Silvio oversees all research, including theory, security and crypto finance.

Silvio is the recipient of the Turing Award (in computer science), of the Gödel Prize (in theoretical computer science) and the RSA prize (in cryptography). He is a member of the National Academy of Sciences, the National Academy of Engineering, the American Academy of Arts and Sciences and Accademia dei Lincei.

Silvio has received his Laurea in Mathematics from the University of Rome, and his PhD in Computer Science from the University of California at Berkeley.