Not Before Time v2.6

Not Before Time is a public broadcast facility built on existing technology. Not Before Time gives three strong guarantees that information:

1. ***will not be readable before a certain future time***
2. ***was not created before a certain past date and time***
3. ***was not electronically signed before a certain time***

These support human rights, democracy and business.

The novel features of Not Before Time are:

1. Time locked information is the opposite of the trend for all information to be available instantly, everywhere.
2. The three guarantees are supplied to ordinary users using their everyday software on their existing devices.
3. The mechanism and its novel new applications is easily explained to non-technical people, perhaps including you.

Version 2.6 3rd April 2023

Dan Shearer

Table of Contents

Introduction and Overview

Not Before Time provides a new certainty to the age-old question "*What did they know, and when did they know it*?"

New fundamental certainties are rare, but there have been some:

> **Reliable timekeeping**. The first portable clocks were developed to help ships navigate. In the centuries since, all of industrialised society has become defined by time. In the 21st century we assume every individual has a right to access accurate time. Legally it is included in other fundamental human rights.
>
> **GPS**. Since the 1990s, always-on knowledge of latitude and longitude has also redefined society. Exact location knowledge about individuals, machines and the urban environment is assumed. Access to GPS is a fundamental individual right protected within the European Union (GPS means "any of the satellite-based location systems".)
>
> **Radiocarbon Dating**. The certainty of C14 dating made history jump into focus. For the first time it we knew for certain when key events happened and what order they happened in. Objects and people could be placed in their correct temporal context. Myths and legends could emerge from the unknown. Humans gained certainty about their past story.
>
> **Evolution**. With the certain knowledge that evolution drives all life, we have begun decoding the petabytes of instructions, historical records and physical potential encoded in DNA. The fundamental certainty is that we *can eventually* decode life. There is no question.

and now...

> **NBT** provides another new kind of certainty. Anyone can connect physical or electronic information with time in very reliable ways.

> *This document is not a paper, and does not cite sources in-line. It is an accessible explanation for how the human needs met by NBT can become a scaleable solution for commercial and community use. See the last section for references and prior art, because there is a technical story too. This document is a conversational approach to explaining why NBT should receive substantial funding.*

NBT is a clock that works for information. Like GPS, NBT is a public broadcast technology that enables new creative opportunities for an unlimited number of users. Like GPS, NBT cannot be limited to people of particular views, nationality, employer or any other difference. NBT removes the ultimate choice about the publishing of information from internet and media companies, and gives it back to individuals. NBT is the opposite of chat with disappearing messages, instead, it is a constant stream of messages from the past appearing in the present.

Technically, NBT contains no surprises. NBT is a mildly different view on thirty year-old technology that is already deployed to billions of users worldwide. There are no new principles or fundamental inventions, only the way they are put together.

NBT resists obsolescence. NBT is intended to be effective for decades and longer despite the fact that decryption techniques get faster and better over time. NBT incorporates state of the art of quantum-resistent encryption methods despite also using thirty year-old encryption principles and includes an upgrade method.

NBT has immediate and scaleable applicability to finance, unlike cryptocurrencies. NBT's application to financial transactions of all kinds including cryptocurrencies is not much discussed in this document. Guarantees of the contents of a time-encrypted finance document has such obvious benefits that it requires standalone analysis. This is just one of the many applications for NBT.

Example NBT Use Cases

- **Embargo.** Secretive Ltd, wishes to make a massive announcement worldwide. An embargoed press release secured by NBT is circulated to press in all timezones, behind all manner of firewalls and on personal devices. Exposure of the announcement is maximised because it is already in everyone's hands, and yet the embargo can't be broken. At the time the embargo expires, the announcement becomes readable everywhere.
- **Sealed bids.** Smart Ltd wishes to build a new office. Commercially sensitive bids are received from many construction companies, with NBT being a requirement for all bid submissions, and bids closing on July 17th 2023. The bids cannot be read before the closing date by anyone, not even the construction companies that sent them. Nobody can accuse Smart Ltd of giving secret pricing information from one bidder to another.
- **Email Decrypted After it is Received.** Using standard Google Email, an encrypted message is received with the subject "This email will be readable from midnight, 25th December 2022", which is in three days' time. Three days later, the email turns to plain text using Google's standard tools for encrypted messages.
- **Whistleblowers.** A whistleblower wishes to publish information but have time to escape. She uses NBT to secure a file which she distributes as an anonymous bittorrent with the filename NBT-Locked-Until-2023-03-01-Interesting-Facts-For-Journalists-Worldwide.pdf.
- **Future payments.** A series of payments can be scheduled in the future by encrypting digital cash with dates falling due at the first of every month, when the recipient will be able to decrypt and use the money. This need not be cryptocurrency, it could equally be an encrypted bank cheque, share certificate or other asset title.
- **Proof of earliest possible creation.** A video is encrypted with a secret password only known to Jemimah, and signed using the NBT for 10am on 11th April 2019, and the video is distributed worldwide. When Jemimah gives the secret password to someone, they know for sure that Jemimah cannot have encrypted the video before that hour in 2019. Nobody can claim they received the NBT-protected video and the secret password in, say, 2017.
- **Physical proof of earliest possible creation.** A controversial paper book is published with a QR code on the front cover. That QR code contains a unique number published under NBT on 18th September 2020. NBT certifies the earliest printing date for the book – for example confirming that it cannot have been printed before certain events happened. This is a form of protection for authors as well as assisting in historical chains of evidence.
- **Unencrypted Email With Proof of Earliest Possible Creation.** A standard Yahoo Email inbox receives a message with the subject "The NBT-signed attachment to email contains meeting notes sent after 14th October 2021". Despite the fact that nothing is known about the sender's email system, and that email is notoriously easy to fake, NBT proves beyond doubt the statement is true. For example, a court of law must reject a claim that the notes were sent on the 10th October 2021, because there can be no doubt about the attachment signature.
- **Combined Creation Proof and Time Locking.** Using standard Google Email, a plain text message is received with the subject "The attached encrypted PDF will be readable from midnight, 25th December 2022, and the first page contains a unique number created on 1st January 2021." NBT makes it certain that the PDF was created after January 2021 and will not be read earlier than December 2022. This is less complicated than it sounds, and has many practical uses.

Technical Principle of Operation

This section requires an understanding of the principles of Public Key Cryptography, for example, as described in https://en.wikipedia.org/wiki/Public-key_cryptography .

There are two phases to establishing a Not Before Time network:

1. Publish a static list of public keys far and wide, with one key for every time interval, for example, every half hour. This means there will be a public key corresponding to 0930 in the NBT timezone on the 3rd February in the year 2043. If the system is intended to last just a modest 25 years then that is 350 000 half-hourly keys, requiring about one hundredth of a second to search on a relatively slow computing device.
2. Broadcast each private key corresponding to the public key at the relevant time. This new private key adds to the growing list of all private keys as time passes, allowing everyone everywhere to unlock information. The guarantees are still kept regardless of any local propagation delays, because it is Not *Before* Time rather than Not *At* Time.

The only significant problem to solve is how to keep the list of private keys secret in a trustable manner, and consideration is given to that later in this document. Everything else is straightforward software engineering.

The rest of this section is an illustration of how NBT can be implemented using everyday technologies, and with the major drawback that this implementation is centralised.

Specification of the Time Format: RFC 3339 describes standard time formats such as can be derived from standard NTP servers including ones fed by atomic clock sources.

Preparation of the Lists of Public and Private Keys: This is a handful of lines of code run on a computer with excellent sources of entropy, calling GPG to generate public/private key pairs for each time interval. The time interval expressed in the selected time format is both the passphrase for the private key and also the name of the public/private key pair. After validation, this yields two lists. The process of keeping the private key list secret in a trustable manner is the main technical challenge for NBT implementors.

Loading the Private Keys into the NBT Server and Starting the NBT Server: The NBT server has an independent and reliable time source (eg multiple atomic clocks checked with GPS). The list of private keys corresponding to the list of public keys is stored within the NBT Server and *only* within that server. The NBT Server then publishes each private key once the time it corresponds to has passed. The NBT Server may be a little late, but never early. The method of publication can include DNS, RestAPI, web page, and more.

Preparation of the Message by an Individual User: A text message is encrypted (eg on a personal computer, with nobody else able to see) using the public key corresponding to the “Not Before” time. This works as follows: the public key is fetched automatically from the NBT server if it hasn’t already been pre-loaded on the device beforehand. The File Save/As dialog box asks for a future time to lock the document, and then encrypts the text using the public key specified by the user.

Distribution of the Encrypted Message by an Individual User: The encrypted message is then distributed using any means. It might be sent as a private email to half a dozen people. It might be put on a public webpage. It might be sent to a server which will wait until the target time has passed, and then email it once the text has been decrypted.

Decryption of the Message by the Recipients or People Generally: The encrypted message can be manually decrypted using any PGP-compatible program once the “Not Before” time has passed. Automated services such as email systems and content management systems can watch the NBT keyserver for the appropriate private key release and then decrypt the message immediately. Proof of concept investigations have shown that building PGP into end-to-end encrypted chat systems is not difficult, especially since cryptocurrency support is becoming more popular in chat.

Technical Description of Guarantees

In the language of Computer Science, the three NBT guarantees give strong cryptographic certainty to the users wishing to share information so that they can:

- choose a date and time before which the information is unreadable, using accepted algorithms and implementations;
- prove that the information was not electronically signed and sealed before a certain date/time (which is new, because electronic signatures do not currently incorporate a timestamp in a universally trusted manner);
- prove conclusively that information was not created before a certain time (which is new, because electronic signatures do not make any statement about the original source of the material they are authenticating)

We can summarise these guarantees in this short form:

1. No Early Reading
2. No Late Denials
3. No Digital or Physical-world Backdating

Real-world Implementation: The Private Key Server

The design described above would not be a good public service. There, the NBT private key server is in principle a single point of failure, either through technical failure or security breach. An attacker with a secret copy of all private keys would be in a very strong position, because if all users trust the system this attacker can quietly monitor for NBT-locked data and decrypt it for their own nefarious purposes.

NBT lends itself to theoretical trust analysis in well-studied fields of computer science, so there is little that is fundamentally new. The main field is called Secure Multiparty Computation, and it addresses the problem of human mistrust.

The human mistrust problem

The fundamental problem is that no one group of implementers can be trusted for NBT. For example if Russian computer scientists deploy the system then Americans will claim that the FSA is in possession of all secret keys, and if the Swiss government sponsors a neutral NBT server then the Chinese government will claim the CIA stole the secret list – and how would we know if that was true or not? If Seiko or some other timekeeping company decide NBT is their commercial future and they wish to lock out other companies from the service, then Apple might issue legal threats to stop them and so on. Nobody trusts anybody else.

The mistrust doesn't just stop with initial key creation. *Even if* the NBT server is set up so that it automatically creates the two lists without human intervention, and *even if* the world can agree on the entity to do that work, someone else will claim that the code involved was compromised (or the computer doing the work was compromised before it was locked down) and so on. And *even if* the notional NBT server was created securely, can we trust where it is housed? We know that if enough money, intellect and political will is focussed then the NBT server can be cracked somehow. Imagine a deceased person is suspected of having created NBT-locked nuclear secrets before they died. The timelocked data might simply be a letter to the deceased's children, but we can also imagine that worried militaries might have a very large budget aimed at cracking the NBT server.

The mathematical solution

Shamir's Secret Sharing (SSS) is a secure multiparty computation algorithm that has been tested and validated for decades, and is trusted about as much as public key cryptography. SSS is accepted to be resistant to even very well-funded modern attackers. All solutions analysed so far for managing the list of secret NBT keys converge to using SSS.

Quoting the non-technical SSS explanation from Wikipedia:

> *The secret is split into multiple parts, called **shares**, which individually should not give any information about the secret. To unlock the secret via Shamir's secret sharing, a minimum number of shares are needed. This is called the **threshold**, and is used to denote the minimum number of shares needed to unlock the secret. An adversary who discovers any number of shares less than the threshold will not have any additional information about the secured secret.*

An even less technical explanation is that SSS is a bit like a horcrux from the tale of Harry Potter, where all of the distributed secrets need to be found before the Voldemort dies.

For NBT, the process of SSS is:

- (say) 12 mutually mistrustful parties each prepare to host portions of the secret keys for publication.
- A one-off process in a single central place creates a list of secret and public keys. This will take a few seconds at most, even if the private keys are for time intervals over the next two hundred years. This requires only a very small amount of code, and can be manually supervised by mutually mistrustful parties.

- a similarly quick process splits each of the secret keys into 12 parts using SSS. These 12 parts are securely transmitted to the 12 prepared servers.
- The central list of secret keys and the full collection of key portions is thoroughly destroyed in a certified way.

There are many implementation details and refinements. But in the end, whether the private key portion servers are co-located with existing time servers, or hosted by various militaries, or launched into space on cheap satellites, NBT always operates in the same way. Mutually mistrustful parties publish their key portions to each other, and anyone wanting to broadcast keys for reading time-locked information must be one of these parties and must collect enough of these key portions to meet the SSS threshold.

Pragmatic Initial Implementation

There are two cases where the problem is easy to solve because security requirements are lower:

1. For an initial commercial live demonstrator. For some of the commercial applications, especially in the legal and construction industries, today's level of security is often so low that a special-purpose NBT would be an improvement even if it involves a vulnerable central server. This would never be good enough for a public service, but it would be a good way of showing how the NBT system works.
2. For a time-limited global demonstrator. This could be the same as the commercial live demonstrator, only that public keys only cover perhaps two years into the future. Again, this less likely to become a high-profile target for attackers since all secrets will be revealed within two years at most.

Technical Steps to Demonstration Projects

The result will be a fully functional prototype by following these steps:

- Create an NBT key server distribution service via protocols such as Rest, DNS, and possibly a private instance of MIT Keyserver.
- Create an NBT client as a web service, which receives unencrypted files and returns them encrypted to an NBT future date, and also receives encrypted files and either decrypts them or states when they can be decrypted. This can be just a wrapper on GnuPG.
- Create a virtual printer device for the Linux and Android operating systems, that functions in the same way as "Print to PDF", except that it asks for a date in the future, and then creates an NBT-locked PDF.
- Create a small modification to LibreOffice, where File/Export As PDF *or* File Save/As has a new field for "Not Readable Before This Date", fetches the appropriate public key from the NBT keyserver, and encrypts the generated file. Libreoffice developers informally estimate just 200 lines of code need changing, and some scripting experiments have been done.
- Create a modified PDF reader *OR* a corresponding modification to File/Open in LibreOffice where if an NBT header is detected then the NBT server is polled to see if there is a private key released. The poll is not necessary if the local time is regarded as accurate.

For production use, there are equivalents to all of the above for most common commercial and open source systems. Intercept vectors such as printer drivers, File/Save and File/Open dialogs are routinely hooked. Closed source SaaS products from Office 365 to Salesforce have large aftermarket communities with addon hooks. Android and iOS development contractors assure me there are multiple ways to solve this without inventing a new class of mobile solutions.

Enabling Existing Email Infrastructure

One early next step should be emails. Currently almost nobody has an everyday motivation to encrypt emails, despite the facts that it improves security and privacy. Even if key management was not a problem, why would anyone bother? Security and privacy are not major motivators for most people including technical people. The GDPR and ePrivacy laws are not exciting motivators for security.

However, emails that are only visible after a certain time... That's instantly graspable by most people. NBT could be a reason for everyone to use the encrypted mail facilities they already have, since the key management problem shrinks to zero in the case of NBT. For many email users, just one simple plugin will give them access to cool NBT facilities.

Of course, even the possession of an encrypted email conveys information, and sometimes that needs to be avoided. It is not difficult to implement an email server that retains on encrypted emails and chat messages until they can be read.

GPS As an Example for NBT Commerce

The NBT broadcast facility is base infrastructure without a direct commercial model, in the same way GPS (and GLONASS, Galileo etc) are not. It does not make sense to try to limit access to the NBT broadcast facility. It took years to realise that applications built on top of GPS was of immense commercial value, including immense value to the preservation of human life and improvement of the human condition. But the base GPS signals for all of the systems have been entirely unrestricted for decades.

For NBT, the immediate monetisable value is in applications on top of this base infrastructure. The human rights use case cannot be shut down without also shutting down commercial activity, so even from launch we have a facility which is a high-value target. The full NBT system needs full SSS.

Related work, and needed work

I have been working on the NBT concept on-and-off for some years, focussing mostly on the societal problems that it could address, and on the transformational nature of a kind of new GPS. In January 2023, I noticed Sufang Zhou et al published the paper Multiple Time Servers Timed-Release Encryption Based on Shamir Secret Sharing . This is a theoretical analysis of a practical real-world implementation of an NBT, with excellent literature research. This paper will make implementation quicker, moving it closer to “just add funding”.

This is about creating a new universal data primitive usable by ordinary people from day one. While this primitive can be incorporated into other work (such as my own on Attribute-based encryption applications) it should be strictly limited in scope to NBT alone.

In terms of solid theory and implementation, the following work is needed:

1. A further paper, narrowing the theory down to the minimum proven or at least well-analysed encryption and algorithms required for a practical NBT network. There are several teams around the world with a likely incentive to collaborate on this. The paper would be a companion to an updated version of this document.
2. An RFC, or RFC-style, definition of the data types and encryption internal to these data types to implement an NBT primitive
3. An RFC-style definition of the protocols to be spoken by (a) nodes who each have a partial key list and (b) end users who wish to access timelocked information

Selected relevant publications:

Rivest R L, Shamir A, Wagner D A. Time-lock puzzles and timed-release crypto [R/OL]. (1996–02–01) [2023–01–15]. http://bitsavers.trailing-edge.com/pdf/mit/lcs/tr/MIT-LCS-TR-684.pdf, 1996

Liu J, Jager T, Kakvi S A, et al. How to build time-lock encryption[J]. Designs, Codes and Cryptography, 2018, 86(11): 2549-2586.

Lai W J, Hsueh C W, Wu J L. A fully decentralized time-lock encryption system on blockchain[C]//2019 IEEE International
Conference on Blockchain (Blockchain). Atlanta: IEEE, 2019: 302-307.

Yuan K, Wang Y, Zeng Y, et al. Provably Secure Security-Enhanced Timed-Release Encryption in the Random Oracle Model[J]. Security and Communication Networks, 2021: 1-10.