Timekoin Server Technical White Paper

Introduction.

This Technical White Paper will step through each part of the Timekoin Crypto-Currency System with detailed explanations about how each aspect of the program works. The document will be broken up into sections, detailing what each file represents and what the intended functions are.

File Sections.
This list is a break-down of every file used in the system as of version 4.02, the finalizing of this document. The list will start in alphabetical order and continue through each file for each section.

Files (short summary):

- **api.php** – Script that allows access to Timekoin like functions from a web address
- **balance.php** – Script that builds a balance index of recently used public keys
- **configuration.php** – Contains database configuration information
- **foundation.php** – Script that monitors and creates Transaction Foundations
- **function.php** – Contains static variables and common functions used by other scripts
- **generation.php** – Script that creates currency or submits request to the peer swarm for generation election
- **genpeer.php** – Script manages and elects public keys that are allowed to create currency
- **index.php** – Script to access web based GUI front-end of Timekoin
- **main.php** – Main Program script that manages the working scripts in Timekoin
- **mersenne_twister.php** – A PHP based version of the Mersenne Twister random number generator
- **peerlist.php** – Script that managers peer connection list and links other Timekoin peers together
- **queueclerk.php** – Script that scoops transactions from the network that should be processed
- **RSA.php** – A PHP based version of RSA encryption and decryption that is more secure than openssl
- **status.php** – Static script that controls what aspects of Timekoin should be disabled or enabled
- **templates.php** – Collection of functions used by index.php for the web based GUI
- **transclerk.php** – Script that monitors and maintains the transaction history for Timekoin
- **treasurer.php** – Script that processes transactions in the peer network
- **watchdog.php** – Script that monitors all other scripts to insure none have become stuck or failed

API.PHP
This script allows web based access to Timekoin like functions externally. This is how the Timekoin Client can send and receive currency without needing the full Timekoin database of transactions stored locally. The server operate sets up a “hash code” that must be included in any query string. It can work like a password of sorts to allow only those you want using it access.

BALANCE.PHP
This script builds a balance index of any used public for transactions within the last 2000 transaction cycles. The purpose of the balance index is to speed up processing of transactions by saving the peer CPU time doing balance look-ups of past transactions. The balance index is also updated when transactions are processed normally if no index exist yet.

CONFIGURATION.PHP
This file is very short, consisting of only 4 lines of code. It contains the configuration that Timekoin uses for the database IP Address (be it local or another server), the Username for the database, the Password for the database, and the actual Database Name itself. This file is meant to be modified by the user to whatever custom needs is required and usually never changes between version upgrades or release.

FOUNDATION.PHP
This script maintains Transaction Foundations in Timekoin. Transaction Foundations are created every 500 transaction cycles (500 x 5 minutes = 1.7 Days) as a way to block up a certain period of transactions into a single SHA256 Hash. The purpose of the Hash is for Peers to be able to compare large blocks of time (1.7 days) worth of transactions for differences quickly without having to compare each transaction one at a time.

Newly created Transaction Foundations (less than 1.7 days old) can become invalid if 2/3 or more of the connected peers disagree with the Hash. When this happens, the Transaction Foundation is marked bad and purged. Timekoin then repairs the area of transactions the Transaction Foundation represented via the other peers and afterwards, attempts to re-create the Transaction Foundation for another peer compare of hash values.

Transaction Foundations older than 1.7 days require 100% of peer disagreement before any purge and repair will take place. Thus, the Transaction Foundations can be thought of as a way to archive Transaction History within Timekoin that can not be modified or changed without 100% peer dis-agreement.

This script is also responsible for creating a new Transaction Foundation when the time is 50 Cycles or 4 Hours pass the next block to avoid peer confusion as the Transaction Foundation is calculated as part of the Transaction History Hash when peers are comparing Hash histories.

The script also does random checks of older blocks with other peers in case of database damage that corrupts transaction data from the past that might need to be repaired.

The *foundation.php* script will respond to hash polls with the SHA256 Hash of a specific block as per this example.

http://timekoin.net/timekoin/foundation.php?action=block_hash&block_number=1
Example output: **789c1568bb57f6b8ee0da12d1d23363b356d2527d5cc5fb0c5f45dfa33dac169**

This is the only communication that the script will have with other peers. It will either poll peers for their hash values or allow other peers to poll itself for the hash value of a specific Transaction Foundation block.

This script is also responsible for updating the Timekoin random seed that is used for random number generation.

FUNCTION.PHP

This file contains static variables such as Program Version and Transaction Epoch, along with other functions that might be shared by other scripts. Documenting each function within this file would be too extensive, so only the major points will be covered here. More technical people can explore the file as everything in Timekoin should be very well documented as to the purpose it performs.

The Transaction Epoch is a static start time that all Transaction Cycles and Transaction Foundations are built against. This can be thought of as a starting clock from which all calculations of time are derived from. During testing, this time was often moved forward with each beta release to avoid complications, but since the initial release of the software, the new time is set and will never be changed now that the network is up and running across the world live.

The *function.php* file also contains static variables for space filler and sha256 testing. These exist to test new Timekoin servers for compatibility with the rest of the network. A Timekoin server that is calculating hash and encryption data differently from the rest of the network would be incompatible. The space filler is used for hash calculations as they do not require encryption data fields. The space filler is actually binary for a word, anyone curious to what it means are welcome to run it through a binary to word converter ;-)

GENERATION.PHP

This file operates the generation of currency and the submitting of election request to the transaction queue. It does not communicate with other peers nor do any peers communicate with it. It works exclusively with the database in preparing election request for generation and building currency generation transactions when elected to the generation peer list. Generation transactions are identical to any regular transaction except that currency is sent from the public key of the peer to the same public key of the peer. Thus it is a transaction from itself to itself. When accepted by the network peers, this is how currency is created.

GENPEER.PHP

This file is a very active communication script. This keeps track of which peers (public keys) are allowed to generate currency, which peers are in the queue for election, and which peers should be purged due to more than 8 hours of inactivity.

This script maintains a “Generating Peers List Hash” which is a MD5 hash of the current generation peer list. This hash is created to allow other Timekoin servers a quick way to compare for list differences. If a list difference is found, each peer will poll the other for the complete list and examine each public key one at a time. The threshold for any list changes is a 51% or higher peer disagreement.

When a new peer enters the Timekoin network and wants to generate currency, it generates an election request and feeds it to the transaction queue. This special “transaction” contains the public key of the peer, along with information that other peers will use (such as domain/IP/sub-folder/port) to do a reverse verification that the peer is located at the IP it claims to be located at. This special transaction is then replicated across all the network peers. The new peer (public key) is then added to the election queue when other peers are able to reverse poll and get a successful answer from the peer that wants to be added to the election queue. During a shared calculated point in the future, all peers begin the election process. If only one peer is in the election queue, it is the automatic winner. If more than one peer is in the election queue, a random selection process begins.

The random selection process grabs the current time and uses it to generate a list of random characters (a – z) and then counts the number of randomly selected characters that exist in the public key of the peer. The peer with the “highest score” wins the election. This pseudo-random selection is necessary to make sure all peers come up with the same score for each key to insure they all “elect” the same peer at the same time.

This script has 3 means of communication. The first is the MD5 hash of it's list. The second is the full list with details such public key, join time, last generation time, and IP address. Finally, the reverse crypt verification string.

To retrieve the current list MD5 hash:

http://timekoin.net/timekoin/genpeer.php?action=gen_hash

Example output: **ca19bc7e5b4c80468940f3d23bd229e8**

To retrieve the full list (50 random entries at a time)

http://timekoin.net/timekoin/genpeer.php?action=gen_peer_list

Which will output text in the following example format:
-----public_key1=<in BASE64 here>-----join1=<unix time>-----last1=<unix time>-----ip=192.168.5.11-----END1
-----public_key2=<in BASE64 here>-----join2=<unix time>-----last2=<unix time>-----ip=172.16.5.3-----END2
-----public_key3=<in BASE64 here>-----join3=<unix time>-----last3=<unix time>-----ip=10.1.10.99-----END3

Which can be sorted out to extract the public key, when it joined the list and the last time it generated currency.

To retrieve the reverse crypt verification string:
http://timekoin.net/timekoin/genpeer.php?action=gen_key_crypt

Example Output: **O3xeKhrLzRn8BCIgxpc1hxfDi9f0vLYNcl8UFVkywpISjdztLeTScQNmr9JpGzaPDSJ1xq**

INDEX.PHP

This file contains all the code that allows the user a GUI web-based driven front-end to Timekoin. This GUI allows the user to start and stop Timekoin processes, change configuration settings, and monitor what is going on in the Timekoin network as well as it's own internal processes through logs. This section will break down each “tab” in the GUI and give a description for all functions and buttons.

1. **Login Screen** – Allows the administrator to login with any username / password desired. Each login attempt has a 1 second delay between tries to help avoid brute force attacks on the username or password. A successful login will immediately send the user to the ***Home*** tab in Timekoin.
2. **Home** Tab – This tab is labeled as the “Realtime Server Status”. It shows the current server balance of *Timekoins* and the reported unix time by the system (called Peer Time).

   It also shows a count down to the end of the current cycle. Each cycle is 5 minutes long and functions as a way for peers to gather up transactions in the network that need to be processed.

   The status of each process of Timekoin is displayed in real-time that lets the user know what Timekoin is currently doing. This screen can also show if any processes has become stalled or failed.

   This tab also functions as a notification of “*Operating in Outbound Only Mode*” or “*Timekoin Might Be Out of Sync with the Network Peers*” when the situation applies.
3. **Peerlist** Tab – This tab is labeled as the “Realtime Network Peer List”. It shows which peers Timekoin is currently linked with. The active peers and reserve peers numbers exist at the top to show active connections and which peers are in reserve should the number of active peers fall below the set threshold by the user. Active peers can be deleted or edit by the user manually. The user can edit a peer and set it to never be removed from the list with the “*Do Not Purge*” setting when doing a manual peer edit.

   The *Show Reserve Peers* button will display what peers are kept in idle standby should an active peer drop off the list due to 5 minutes of inactivity.

   The *Add New Peer* button allows the user to manually input peers.

   The *First Contact Servers* button will allow the user to both view and edit the list of servers Timekoin tries to contact first when it is started (or restarted). The First Contact Servers are run by volunteers and exist in a static location on the Internet as a way to get new Timekoin peers into the swarm network faster by providing larger lists of peers willing to link with in Timekoin.
4. **Transaction Queue** Tab – This tab labeled as the “Realtime Transactions in Network Queue” shows a real-time view of the transaction currently awaiting processing for Timekoin. Transactions can range from regular spending transactions from peer to peer, to election request for currency generation, then to actual currency generation transactions when currency is about to be created.

5. **Send / Receive** Tab – This tab will show the user the current public key belonging to the peer and also contains a section to spend Timekoins to the public key belonging to another peer. A short message can be encoded into any transaction for the recipient of the public key.

   The Easy Key website service is also integrated here to allow users to create simple shortcuts for long public keys that are easier to remember or send in messages.
6. **History** Tab – This tab labeled as the "Transaction History" shows a history of transactions of Timekoins either sent to the peer or sent from the peer. This section will also show any encoded messages in the transactions.

7. **Generation** Tab – This tab labeled as the "Crypto Currency Generation" will show the user the current stats for currency generation ranging from production time to current status and amounts generated per cycle. The various buttons for "*Enable / Disable Generation*", "*Show Generation List*", and "*Show Election Queue List*" are self explanatory. The Generation IP field contains the IP address that the peer will broadcast to all peers when generating currency or trying to get elected to generate currency.
8. **System** Tab – This tab labeled as the "System Settings" will show the user buttons to start / stop the Timekoin processes, and custom settings to allow the user to tweak how peers are handled.

   Maximum Active Peers is the most peers that Timekoin tries to keep connected at a time automatically.

   Maximum Reserver Peers is the most peers that Timekoin keeps polls for active connections in case any of the active peers are removed from the list due to inactivity.

   Server Domain is an optional domain that other peers can use to connect back to the Timekoin server with. This can be useful for websites running timekoin as other peers will be able to find it via the domain name rather than a raw IP address.

   Timekoin Subfolder is the folder path relative to the peer IP address or Domain. This is where other peers will find the scripts to connect to for peer exchange or information exchange for transactions.

   Server Port Number is the port number that other peers will connect at. By default, Port 80 is used as it is the default for web servers, but this field can be changed if you want to run a web server on a port other than 80 and allow peers the ability to connect inbound properly (port 8080 for example).

   Max Peer Query is the maximum number of request that any single peer may perform in a 10 second period. This feature helps to defend against flood attacks from either malfunctioning peers or attacks directed at the Timekoin server itself. IPs are banned for 24 hours unless the user manually stops and restarts the Timekoin process in which the IP ban list is cleared.

   Allow LAN Peers is a feature that will turn off LAN IP filtering. Normally Timekoin tries to filter non-routable IP address (such as 192.168.1.X range) from the peer list as they can not be communicated with across the Internet. The purpose of this feature is for those that run a large cluster of peers within a LAN environment and would rather Timekoin manage peer connections rather than manually entering peer information for LAN peers.

Allow Ambient Peer Restarts is a feature that allows random polling from other peers to re-activate a failed Timekoin process or server. Each time a peer polls a Timekoin server with this feature enabled, it does a random check of the Timekoin process to see if it is still running. If the process is found to be failed for any reason, it will attempt to restart the process. This is useful for servers where the web server itself goes through many updates or restarts that interrupt or stop the Timekoin process (such as Leap Second Time Updates or Software Updates)

Check Peer Clock & Ping Times button will begin a poll process of all active peers to see how it's Internal clock compares to the clock of the other peers. This is a useful diagnostic tool to determine if the Timekoin server clock is too far out of sync or if certain peers are too far out of sync with the other peers in the network.

The **Miscellaneous Server Information** section contains technical information that can be useful for diagnostic reasons within Timekoin.

Generating Peers Lish Hash – The current MD5 Hash of the Current Generating Peer list.
Transactions History Hash – The current MD5 Hash of the Transaction History, built from the last Transaction Foundation SHA256 Hash and all the Current Cycle Transaction SHA256 hashes.
Transaction Queue Hash – The current MD5 hash of all Transactions in the Transaction Queue, built from all the data in the transaction such as timestamp, keys, and encryption data.
Transaction History Records – The total number of records in the Transaction History.
Transaction Cycles – The number of 5 minute transaction cycles since the Epoch start time for Timekoin.
Transaction Foundations – The number of Transaction Foundations since the Epoch start time for Timekoin. Each Transaction Foundation is built every 500 transaction cycles.
Uptime – The amount of time that the Timekoin process has been running either from the user starting the process or the process being restarted by automated means.
Database Size – The current size of the entire Timekoin database including all logs, transaction history, and options or peer related data.

9. **Options** Tab – This tab labeled as the “Options & Personal Settings” allows the user the ability to change the username and password used to login to the Timekoin web based GUI front-end. This section also allows the user to change the refresh rate of active pages and set an external hash code.

   The “**Hash Code for External Access**” is a simple code that allows other programs outside access to certain functions within Timekoin such as finding the current balance for a specific public key.

10. **Backup** Tab – This tab labeled as the “Backup & Restore Keys” allows the user the ability to make copies of the private and public key or restore/overwrite the current private/public key with a new one. This serves as the best backup ability for Timekoin as a complete server failure can be restored with only the private and public keys.
11. **Tools** Tab – This tab labeled as the “Tools & Utilities” allows the user to check logs or perform check / repair functions on the transaction history database in Timekoin.

    History Walk – This button allows the user to start anywhere in the transaction history and do a 500 cycle forward walk of transactions to check for timestamp and hash consistency. It can be used to spot damaged transaction blocks of time for example.

    Schedule Check – This button allows the user to schedule a transaction check from any point in the past. The checks use other active peers to compare data with for damaged or incorrect transactions and attempts a repair based off of this group data. Each scheduled check will do 15 cycles forward of checking. So for example, if you start at block # 20,000, it will check 20,000 and 14 more after, stopping at 20,014.

Repair – This button will force a forward repair of all cycles from the point specified by the user all the way up to the current time. This will only repair calculated hashes and will not affect transaction data. The usefulness of this utility is to fix broken or mis-calculated hashes within the transaction history.

Show Last – This button in conjunction with the "*Filter By*" drop down list allow the user to check the logs for any useful information such as generation events or failures of other peers that are logged. The logs can also be cleared with the "Clear All Logs" button to either save database space or start a fresh log when troubleshooting any issues within Timekoin.

12. **Logout** Tab – This will clear all cookies and log the user out of the Timekoin GUI.

MAIN.PHP

This file is the main event firing processor for Timekoin. It's purpose is to run an infinite loop that will activate other scripts in order so that each can process different task in Timekoin (peer management, transaction queue, etc.) all independent of each other.

This file also does random checks to timekoin.net for a time reading. If the internal clock is too far off from the timekoin.net reading, displays a warning for the user in the web based GUI.

Another responsibility of the file is to check on the peer IP usage to determine if an IP should be banned for flooding with queries or an attack.

Each task done by the main.php file will reference the database to determine the current program "state". The active state of 0 means that it is inactive and will not run any other scripts. An active state of 1 represents an online state that is ready to process. An active state of 2 means that the file is currently running in memory and thus processing commands and other scripts. An active state of 3 means that it is time to shutdown this file and revert back to state 0 for an offline mode. No further processing will take place. A table arrangement below will outline how this file processes Timekoin for a better visual representation.

Main.php Process Flow

Database Values from "main_loop_status" table	***Description / Purpose***
main_heartbeat_active = 0	No other script processing, awaits activation from user
main_heartbeat_active = 1	Script is now active, awaits for process to begin loop
main_heartbeat_active = 2	Script is working, such as activating other scripts and features coded into the file.
main_heartbeat_active = 1	Script is finished, but still active, awaits the start of another loop
main_heartbeat_active = 1,2,1,2,1,2,1.....	The cycle repeats over and over
main_heartbeat_active = 3	The script will revert to active state 0 when finished and no further processing will take place. This is the same as a shutdown mode that is called from the database.
main_heartbeat_active = 0	The script is now offline

The database also contains a value for ***main_last_heartbeat*** as this value is recorded for the time when each process loop finishes. The purpose of this value is to determine if the process is running properly or not. When the main.php is running properly, it will update this field every +10 seconds with a new timestamp. Since each timestamp is just a record of the current unix time, it's easy to compare the heartbeat time to the current time and determine if the process is running properly or has become stuck or failed.

This is the same basis that all scripts are compared against as each script reports in the time that it finishes. Any script is considered failed / stuck when it takes longer to report the finish time than the allotted time that has been set for the script to run (maximum processing time). This max time is a safety setting to prevent any one part of Timekoin from going out of control (infinite loop or code error) due to some unknown bug or exploit in the program.

MERSENNE_TWISTER.PHP

A php based version of the Mersenne Twister Random number generator. This is how all Timekoin servers produce the same Random numbers in lock-step so that Peer Elections and Currency Generation take place at the same time around the world regardless of timezone. The seed for random number generation is taken from a recent Transaction Foundation where the Hash is scored out into a random number to feed into the random number generation. Basically, every transaction that goes into Timekoin changes the future for how currency is created and how peer elections take place and are scored.

PEERLIST.PHP

This file will manage both the active peers and reserve peers in Timekoin. When Timekoin is started, it will pull from a “First Contact Server” list of peers that are active and available 24/7 on the Internet. From there, it will seek out more peers and reserve peers to connect with.

As an additional task, this file is also responsible for the random ambient peer checks feature which works by doing random checks of the *main.php* Timekoin process when polled by other peers. Anything that has terminated the Timekoin process, this script will be able to restart it.

The next parts will detail how this script communicates and locks communication with other peers.

1. After the First Contact Servers are added to the active peer list, additional peers are polled from them and added to the reserver peers table.
2. First, Timekoin attempts a simple CRC32 ping to make sure the other peer is running Timekoin in the correct IP/Domain location by sending a poll with a randomly selected number from 1 to 999,999 using this type of poll example:
   http://timekoin.net/timekoin/peerlist.php?action=poll&challenge=5
   Which would output: **b0460072**
3. With a valid match response, Timekoin knows that this peer is active and online answering queries. From which point, Timekoin will send another query to the peer asking to Join.
   Example:
   http://timekoin.net/timekoin/peerlist.php?action=join
4. The peer can give 3 answers. “OK” means it is willing to accept the peer. “FULL” means the peer is at the user set maximum peers and no more are allowed to join with it. Finally, no response can mean a transmission error or some other reason why the peer is not accepting connections at the moment.
5. With a successful “OK” message, Timekoin now sends an “Exchange” query that ask the other Peer to add it to the peer list and in turn it will add the new peer to its own peer list for communication.
   Example:
   http://timekoin.net/timekoin/peerlist.php?action=exchange&domain=timekoin.com&subfolder=timekoin&port_number=80

6. When the other peer has successfully added it to the list, it will report back a status message along with information the other peer needs such as domain, sub-folder, and port information.
   Example:
   -----status=OK-----domain=timekoin.com-----subfolder=timekoin-----port_number=80
   or if a failure occurs
   ----status=FAILED-----domain
7. Then the exchange is finished, both peers now have each other's information for which further communication can take place.
8. Each peer sends a random poll request for each active peer to make sure they are still answering queries. Any peer that does not answer poll request for more than 5 minutes is considered offline and is purged from the active peer list.
9. The script is also responsible for reporting the local clock time (unix time) when polled as the example below:
   http://timekoin.net/timekoin/peerlist.php?action=polltime
   Example output: **1344630265**
10. When one peer is polling another peer for a list reserve peers to add, it uses this query example:
    http://timekoin.net/timekoin/peerlist.php?action=new_peers
    Example Output:
    -----IP1=-----domain1=timekoin.com-----subfolder1=timekoin-----port_number1=80-----
    -----IP2=172.18.14.33-----domain2=-----subfolder2=timekoin-----port_number2=8080-----
    -----IP3=-----domain3=timekoin.org-----subfolder3=bobkoin-----port_number3=8888-----

QUEUECLERK.PHP

This file manages transactions in the network that need to be processed at the end of the current transaction cycle. This is accomplished by polling all active peers for their own transaction queues and making a copy of any transactions that do not exist in its own queue. To speed up the process without having to download the entire queue of every peer, each transaction is referenced by its SHA256 hash. The entire process will be detailed in the section below.

1. First, each active peer is polled for their entire queue MD5 hash with the following query example:
   http://timekoin.net/timekoin/queueclerk.php?action=trans_hash
   Example Output: **9ec0209b67d7e18bce0cdd322258d917**

2. This is compared to what the peer has calculated for its own transaction queue. If the numbers match, then both peers have identical transactions in the queue already and there is no need to exchange transaction information.
3. When a difference is found, the next step is to request a list of every transaction SHA256 Hash with the following example query:
   http://timekoin.net/timekoin/queueclerk.php?action=queue
   Example Output:
   ---queue1=a7921ff2a4d3257783051776756a8f0f8da6d3507fa7c1441f13198dd265949e---end1
   ---queue2=30df141a42811c6b48a07249949722a6038234a8ead8fa0f455ba9e849331a5f---end2
   ---queue3=1c9db508b5b55fc140878bcbe6347de9b6bd1f49b34016533b9ec474e8f77ee7---end3

4. This list is then compared to it's own list. When a queue item is found that does not exist in its own queue, it will now request the full details of the missing transaction with the following query example:
   http://timekoin.net/timekoin/queueclerk.php?action=transaction&number=a7921ff2a4d3257783051776756a8f0f8da6d3507fa7c1441f13198dd265949e

5. When the query sent, a long block of data is sent back containing all the transaction details.
   Example Output Below:
   -----timestamp=1344632108-----
   public_key=LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlIZk1BMEdDU3FHU0liM0RRRUJBUVVBQTRITkFEQ0J5UUtCd1FERldCMUd0RlFNaForN3NWcysrRkRZTlRBVApETkJKYVhVdkoycEpIRGV2YUx3MDZWOUR4ZDhqYVJHYTZqOE16VW84OE9oVmRBeHdiWlVWaDZQeVhTVEFUbEoxClY3aHY1d2loM0Y1cjhBYXBWbTgwN0xVQkc4eDFuemhKR1VGM1o5bnRDcXJDK1J1cFdFR05ZbC9aaWJVUjNxWjkKSWttUWttbktQR1BCajY1RmV4TGpRdnBsam10SFEzVzlnZlNpQ3FSWUpwanoyUFQxQWpkaWhXVWJ2Q0xkQzUybgp3em12VmJvMDFFdHN6K0NJdVV3MW1JbUx6MEVQSHRWcVBCOVorbUVDQXdFQUFRPT0KLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==-----
   crypt1=xI2HvwIGAIM4Ccdt5avlQJk/
   XTB9pZB37ZsWuS+otoeVBGw1yBALl+azlZOhEhJsiUOSjN+kDVq06fCLAdaHaaym648p1EnP+kG+FAc48MiBgl7GXcA
   GsIz19xR1ad5JpG5vL35zZoJSHz8z7ZhZmczvst6xZRoIlwcZStPvyX5KuZnWkLfogEgUW3OdrX/
   veN2PYUg+s7eVRzNxjDsraNNbRIIfxdpqf7e94f9OO9orn34VsBfgPAsuaYpopo03-----
   crypt2=JoXjxxGRu0vZaTA3eq87fV7iQyubRrlroUUXKvvpkSr8GG1dDOLpubJCSre/DJe6fp/
   lDmTb1k3IXg5dnkBox0a1EYHCO0Ai6ZDRiVU710I0WN2ZsCtL+iEO4kzC3m/
   Ynd8d6HrV9bTFmXAkPFFUAET1J8OrVz6tscjXahb8RiTh/
   wGnfJmA8suvZN5A1Qo4nZuMtji9E4kQmNArwUZXsAo4/5ZDN1lCABQcDimdlPiN85nLVCAsMczuQnLFJFt9-----
   crypt3=VB0Sh8FvjqD4Ms+
   +n0iJxxIo536BWISFqQ+02sSvBTUsLtpYEiYH1WTRrdwmpzFB5HHOQAjgOdv41uqbjsvzSm4i66uL3DZzE43a0lxeXgQyz
   KWH1M/NHm82Rdo7LUQJ0olWj/KpDUQtJDlc46QpBrnfd/zeVlpAchiuOL7Wip9VnxBoQQllSvP3kMhD1I/
   luOLdCbOTxQcUp8An3qE4ogHTAVMVI0gxEJ/l7Fz3j+5dWzTy3E27RIiY02MykIPn-----
   hash=6cfb66a7340c4df686dcc5a16d29ebd41bb2ad6de585cac5a7fc0d884cc7aeb1-----attribute=G-----end---
   qhash=d898ddfdd27377393f9172cb201f7bf0---endqhash

6. Timekoin now compares the transaction data to what is already in the queue. Duplicates are discarded and transactions with invalid or corrupt data are also discarded from being added to the queue. Timekoin also checks for a limit on transactions from a single peer of 100 or less. Anymore and the transactions are ignored to avoid any flooding that could take place in the network.

7. This script will also accept inbound transactions that are posted from external IPs. The purpose for this is to allow peers trapped behind a firewall to still be able to communicate out transactions they want the Timekoin network to process.
   Example post transaction:
   http://timekoin.net/timekoin/queueclerk.php?action=input_transaction
   Post Data Fields:
   [timestamp][public_key][crypt_data1][crypt_data2][crypt_data3][hash][attribute][qhash]

8. This script also has a external function via the server hash code for getting the timekoin balance of any input public key via this example query:
   http://timekoin.net/timekoin/queueclerk.php?action=key_balance&hash=abc12345
   Post Data Fields:
   [public_key] = Encoded as Base64
   Output will be the current balance of that public key.

RSA.PHP

A php version of the RSA or (Rivest–Shamir–Adleman) public-key crypto-system. The default implementation through OpenSSL has issues with encryption or decryption when using odd, non-standard key sizes. It can also produce very weak Private Keys. All Private Keys in Timekoin are created through this custom implementation of the RSA encryption to increase their strength against key cracking. This also allows Timekoin to scale encryption on the fly so that it can always stay ahead of Quantum computers that one day might break weaker encryption levels.

STATUS.PHP

This file serves as a way to disable all Timekoin processing scripts or disable only a single script from processing. This is useful for debugging problems in Timekoin or disabling the whole system in case of an emergency. The defined variables are written into this file so that even in the event of a database failure, Timekoin can be disabled by the user.

This is the default code in the file for reference reasons. 0 means the script or system is allowed to process. 1 means the system or script is disabled.

```
<?PHP
define("TIMEKOIN_DISABLED","0");
define("FOUNDATION_DISABLED","0");
define("GENERATION_DISABLED","0");
define("GENPEER_DISABLED","0");
define("PEERLIST_DISABLED","0");
define("QUEUECLERK_DISABLED","0");
define("TRANSCLERK_DISABLED","0");
define("TREASURER_DISABLED","0");
define("BALANCE_DISABLED","0");
?>
```

TEMPLATES.PHP

This file serves as a output template for the **index.php** file that runs the web based GUI for Timekoin. This file formats and outputs the structure for the GUI as well as process stats for various parts of the Timekoin GUI such as Database Size, Configuration Settings, etc.

TRANSCLERK.PHP

This file serves as the transaction history processing script. The purpose of this script is to monitor and maintain a real-time transaction history for itself and other peers that are linked with Timekoin. This script is also responsible for repairing any damage in the transaction history and doing random spot checks of the transaction history via other connected peers. This script is also responsible for initializing a new install of Timekoin when the database transaction history is blank.

This section will cover how a new database is setup and what process is used to compare and repair the transaction history of Timekoin.

1. A new install of Timekoin has a blank transaction history table. The first time Timekoin is run, a single transaction is added to the database table. This transaction has the special attribute of “B” which stands for the Beginning Transaction. This transaction starts with the global Transaction Epoch timestamp and writes out a static transaction of binary code and a binary hash. The purpose is to create a single transaction that future transactions can build a hash chain from.
2. Afterwards, 3 more Hash transactions are written to the database using the first one as a building block for the others. After the first 4 records of the table are complete, the last record is compared to a static known SHA256 Hash recorded in the **function.php** static variable list.
3. This test is done to make sure the peer is processing the encryption and hash generation properly. If the results do not match what is set in the **function.php** file, then the peer can not participate in the Timekoin network due to differences in how it will process encryption and hash data that is incompatible.

4. Once this test has passed, the script will switch to a “live” database where it will begin catching up on past transactions using it's current active peer list as the source for history restoration.
5. As part of any repair process, this script has certain functions that it can access from other peers and vice-versa from itself.
6. First Timekoin will poll the History Hash of all the peers. This is done by the example query:
   http://timekoin.net/timekoin/transclerk.php?action=history_hash
   Output Example: **7aac4329af89f549b6444b872e00132c**
7. This MD5 Hash is built by using the number of records in the transaction history table, current Transaction Foundation SHA256 Hash and the all the most recent Transaction SHA256 Hashes that exist from the beginning of the current Transaction Foundation Cycle to the present time. This gives Timekoin a quick way to compare if it contains a transaction history which also matches all of its own peers.
8. When a difference is found, Timekoin begins comparing individual transaction cycles by starting with the transaction cycle SHA256 hash for a specific transaction cycle. An example query below:
   http://timekoin.net/timekoin/transclerk.php?action=block_hash&block_number=0
   Example Output: **0d0492b0a84750ef6c733a97a050749325a7f5570bd04234d02ac23668feb013**
9. All peers will report their own SHA256 Hash for this transaction cycle and Timekoin will build a comparison list. When nothing exist to compare against (new database), Timekoin then relies on the majority to decide what data should fill the transaction cycle.
10. To grab a copy of this transaction data, Timekoin will send this query to all active peers for the full dataset which includes transactions, generation, and hash events.
    http://timekoin.net/timekoin/transclerk.php?action=transaction_data&block_number=13
    Example Output:
    -----timestamp1=1338580200-----
    public_key_from1=MDExMTAxMDAwMTEwMTAwMTAxMTAxMTAxMDExMDAxMDE=-----
    public_key_to1=MDExMTAxMDAwMTEwMTAwMTAxMTAxMTAxMDExMDAxMDE=-----
    crypt1data1=01110100011010010110110101100101-----crypt2data1=01110100011010010110110101100101-----
    crypt3data1=01110100011010010110110101100101-----
    hash1=ac9517d0b90f83d70ff19bc5359cfae82aaafacf20a77420818db6397fefee20-----attribute1=H-----end1-----
    timestamp2=1338580241-----
    public_key_from2=LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlIZk1BMEdDU3FHU0liM0RRRUJBUVVBQTRITkFEQ0J5UUtCd1FEYkVDWEFNZE96YW45ajNRazdNdUhJbVQ4VAoycnlHODdTUXFuNWRxZVArNGVtY3FxOWlqMlZWYjl5T1N0QVJQaVJ3dDRxUlc5TlpvT0FZdjNvbUllUFYzc2ZmCmNaWkdDTWs4QmVaSVRYeDNrWWhNN0tkeS96RUNhbTRMV1JyR29wZS9sVTlxMlg2S21nNWVCSndYbURFZVFwdHIKVnNRd2VOclpZRkNyekZqUXN1VnByL0I1REEwNERhTkhkOFovMzRLRkxPSDh2N0JqeVZNK2NKN0p0VkhZUjQvKwp5VDIxUTJUZmxUSk1CWWY3djJYMGhBdHh1Zk44clBhK2tBdGhiZVVDQXdFQUFRPT0KLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==-----
    public_key_to2=LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlIZk1BMEdDU3FHU0liM0RRRUJBUVVBQTRITkFEQ0J5UUtCd1FEYkVDWEFNZE96YW45ajNRazdNdUhJbVQ4VAoycnlHODdTUXFuNWRxZVArNGVtY3FxOWlqMlZWYjl5T1N0QVJQaVJ3dDRxUlc5TlpvT0FZdjNvbUllUFYzc2ZmCmNaWkdDTWs4QmVaSVRYeDNrWWhNN0tkeS96RUNhbTRMV1JyR29wZS9sVTlxMlg2S21nNWVCSndYbURFZVFwdHIKVnNRd2VOclpZRkNyekZqUXN1VnByL0I1REEwNERhTkhkOFovMzRLRkxPSDh2N0JqeVZNK2NKN0p0VkhZUjQvKwp5VDIxUTJUZmxUSk1CWWY3djJYMGhBdHh1Zk44clBhK2tBdGhiZVVDQXdFQUFRPT0KLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==-----
    crypt1data2=k6rdl34DsacGwh4qJx5GC8v39hLL0Y0qCEQfjnUfcBsF7xPBOgHcnJyRPmI7vT8TsWRaD/
    CE7Of7c1Lf7uULB99zQrUdY/fnxLAoBtVgQG0hI2KF/9kWZvDbVp4v8sUHiPJ1tGKL6S+H/
    Un61I6lCaPGyiLkyDi8d2NO4tImTVPDatt8sZxZx69tJO098QRDMwBM1vS/
    4t9SXKpxAvTWzSZ4NaJ67PYXRevl7akqeOzH/zeYmCA5mNFRfv13IcAT-----
    crypt2data2=agCozZpKzb57k3IB9PHFC8nl02Q9T9zY9KOSra1hnWD8zfQFKWGN22S8iqLRhszjzCBconeJw6Wu6fHmOrzBhmTYTa3HyX3YP+uXFb8+2X8V4ncR9/
    iwL2U58E2WGxZqnwbpbF7P7p6wPZ5fnkO1au44Dn6CGxLCkca4pPd8mh3r7WdVIQWS3Lrw8RcoLT4UYgqPM9i+m9cFDnzg1gJy3QBmah8MjeGFxt/cpOSLvcjb1UAyD9SUzZVHc6w2XM8w-----
    crypt3data2=hKuKiXIwUhurj1e5HenbOeRj7e2IxnkEJ5t3uFJ8UzvPMup8LdxmUUJcoq0irJecTnVJo7qRaRT3wKes7+uM9cpurTpUfF+q2W6X50kfsYWcpM9F0nP3SbGJfp60wp6R88is/DVC/
    5Iq59GuyPQCbeLSH+qCJK6Gi86vBdkfknMlUbKxqTHzpngr5GTtTI18uDZSSBtZNRDaW0mj1UUcV0rpGYfn/
    qZsVRJgGuXFm7P/0dtXCWxCNQTVWrR81Tdn-----
    hash2=40f6fb411a4ab4bafe713d4faa63b127e6e43c917a1a5403f59263a27e32eb75-----attribute2=G-----end2

11. Afterwards, the data is copied into the database transaction history and then a double-check is performed by taking the actual data and building a SHA256 hash to compare to the SHA256 hash from the peers to check for data tampering.
12. A 51% or higher majority agreement among active peers is required for the new data to remain in the database. Otherwise, a peer conflict occurs. When a peer conflict occurs, it means that one peer has transaction data for a specific cycle and another peer has different data for the same transaction cycle. This cycle of conflict is resolved by contacting different peers and examining what data they might have for the same transaction cycle. The data is sorted out so that the majority peers (51% or higher) with the same data is considered correct.
13. When there is no majority of data agreement among peers, the transaction cycle is logged as “Too Much Peer Conflict for Block #” in the log file. The transaction cycle is left blank and the transaction cycle afterwards is then put in line next to be examined. The transaction cycle with too much conflict is left blank, but will be revisited again in the future to re-check for peer agreements on what the data should be.
14. Even with a 100% network sync transaction history, Timekoin continues to do random transaction cycle checks of the database on a regular basis to spot tampering, corruption of data, or missing data to be corrected. This random spot check starts at the beginning of the current transaction foundation cycle and continues to the present time. When a transaction cycle appears to be incorrect, Timekoin will double-check the accuracy of the data with other active peers for verification and those peers will also perform verification testing as well.
15. This constant cycle of checks is what maintains the accuracy of the transaction history as the effects are multiplied as more and more peers join the network doing more and more random transaction history checks and communicating this back to other peers to verify.

TREASURER.PHP

This file is responsible for processing the network transaction queue, generating the SHA256 hash of the previous transaction cycle, and processing any transactions that exist in its own personal transaction queue from the user. This is where a lot of the security features of Timekoin come into play as it uses the encryption data and SHA256 hashes to do balance checks and filter out any bogus or corrupt transaction data before it is stored in the database transaction history table.

The only communication that this script will perform with other Timekoin peers is submitting transactions that should be placed in the network queue for the other peers. This script has no inbound query functions that other peers can use. The whole transaction process is better detailed near the end of this document.

WATCHDOG.PHP

This file functions as a process monitor for the other scripts. Should any of the other scripts take more time to process the task than what is allowed, the watchdog will attempt to reset their status in the database. Other scripts that encounter some unknown bug or exploit can be restarted this way by the watchdog and the problem recorded in the log files for the user to examine later if the problem becomes a major issue.

This script is more useful for very low end systems that don't always have the processor power or resources to complete task within the default time limits set in Timekoin.

Transaction History Detailed

The exact workings of the transaction history is detailed in this section. To begin to understand how this is able to process and secure transaction, you must start with the concepts of what Timekoin is using as far as encryption and hash verification.

Timekoin uses Asymmetric Encryption as part of the transaction data. The private key and public key are generated at the same time when installing a new Timekoin server. The private key is kept with the user and the public key is available to the public. The private key is used to encrypt data and the public key is used to decrypt the same data.

Because only the private key can create valid encrypted data that only the public key can unlock, no one can create false data that pretends to work with the same public key. This allows the user to create a transaction that states a certain amount of Timekoins should be given to another peer (or user) via their public key.

Timekoin uses 1,536 bit key sizes by default. That makes the private key about 1276 character long and the public key about 360 characters long. When a user creates a transaction, the data inside the transaction contains the destination public key and the amounts that should be sent to it.

Here is an example transaction and how it is built.

First, it starts with the timestamp. This represents when the transaction was added to the network queue to be processed by all the other peers. The timestamp is Unix time based, just an integer number. The next field is the public key that will be used to unlock the encrypted data. The next 3 fields all contain encrypted data. The encrypted data contains the public key of where the Timekoins should be sent to as well as the amount, time that the transaction was created by the user (not the time it was put into the network queue) and a SHA256 hash of the first 2 encrypted fields.

Finally, a SHA256 hash is created of the 3 encrypted fields, and the last field is the attribute for the transaction.

Example:

timestamp	public_key	crypt_data1	crypt_data2	crypt_data3	hash	attribute
1338580241	-----BEGIN PUBLIC KEY----- MIHfMA0GCSqG SIb3DQEBAQU	k6rdl34DsacGwh4 qJx5GC8v39hLL0 Y0qCEQfjnUfcBs F7xPBOg	agCozZpKzb57k3 IB9PHFC8nl02Q9 T9zY9KOSra1hn WD8zfQFKW	hKuKiXIwUhurj1 e5HenbOeRj7e2Ix nkEJ5t3uFJ8UzvP Mup8Ld	40f6fb411a4ab4ba fe713d4faa63b127 e6e43c917a1a540 3f5	T

The reason that the encrypted data is broken up into 3 fields is due to the limited size of the encrypted blocks. Each block can only have 181 characters encrypted into it with a 1,536 bit key. The need to encrypt the public key into the transaction means it must be split in half and each half encrypted with the private key. The 3rd encrypted field contains the amounts, timestamp of when the user created the transaction and a SHA256 hash of the first two encrypted fields, along with a short message in plain text if one is included by the user. The 3 encrypted data fields are converted to base64 to allow easy database storing and retrieving of the data as well as transmission across the Internet.

Using the example chart on the above, if you removed the encryption layer, the same transaction could look like this example below.

timestamp	public_key	crypt_data1	crypt_data2	crypt_data3	hash	attribute
1338580241	-----BEGIN PUBLIC KEY----- MIHfMA0GCSqGSI b3DQEBAQUA -----END PUBLIC KEY-----	-----BEGIN PUBLIC KEY----- 2ryG87SQqn5	CZZGCMk8Be -----END PUBLIC KEY-----	AMOUNT=5 ---TIME=1338580201 ---HASH=87c2dd3adf08e ---MSG=hey there	2ba73d90155f5	T

When it is time for the *treasurer.php* script in Timekoin to process the network transaction queue. It starts with the attribute type. Only two types of transactions are currently processed. Those that have a **T** or **G** for the attribute.

The **G type attribute** represents currency generation. This type of transaction can only be processed during the random allowed time that all peers agree to. During this small time window, only the public keys in the generation list are allowed to create transactions that give themselves currency. The script checks that the transaction amount is within the defined rules of generation time (the longer the peer has been generating currency, the more that will be allowed to generate)

The time for allowed currency generation is pseudo-random generated based on the next transaction cycle time. This allows all peers to "generate currency" at the same time without any need for a special agreement mechanism. The forward movement of time is the deciding factor for all of them.

The rules for maximum generation amounts are defined as such below and every peer must follow these rules or the transaction will be discarded as invalid by all peers.

1. 0 to 1 week – 1 currency unit per generation cycle
2. 1 to 2 weeks – 2 currency units
3. 2 to 4 weeks – 3 currency units
4. 4 to 8 weeks – 4 currency units
5. 8 to 16 weeks – 5 currency units
6. 16 to 32 weeks – 6 currency units
7. 32 to 64 weeks – 7 currency units
8. 64 to 128 weeks – 8 currency units
9. 128 to 256 – weeks 9 currency units
10. 256 or greater weeks – 10 currency units

The **T type attribute** represents a regular transaction between two peers (with public keys). When a transaction of this attribute is processed, many layers of security must be peeled away.

1. First, the transaction sender's public key and transaction hash are checked for duplicates in the database. All Timekoin transactions are unique and even if sending the same amount to the same person (public key) will always generate a different SHA256 hash due to the time difference encoded into the transaction itself. For this reason, no duplicate transactions are allowed from the start.

2. Next, the public key is used to unlock the encrypted data. The encrypted data is sorted into these fields. **[destination public key] [amount sent] [sha256 hash of destination public key]**

3. Next, the amount sent is checked against the database to see if the public key "sending" the currency actually has enough balance to make the transaction. This prevents any peer from spending more than it has for a balance. Often referred to as a "double spending" prevention check.

4. Next, the 3 encrypted fields are checked for tampering by building a SHA256 hash out of the data from all 3 fields and comparing it to the SHA256 hash that was sent along with the transaction. If the built hash and the hash sent with the transaction match, the transaction will be recorded into the transaction history table of the database and considered complete.

5. Last, you will notice nothing was done with the encrypted SHA256 hash in the 3rd encrypted data field; why is that? This field is actually used to screen out invalid transactions before they even arrive in the transaction queue for Timekoin. The *queueclerk.php* script already performs this scan on all inbound transactions that are being inserted into the queue by extracting the SHA256 hash from this field and using it to compare if the "destination public key" from the 1st and 2nd field for the transaction has been tampered with or modified from its original value. For this reason, it did not seem logical to process the same security feature twice since modifying any of the 3 encrypted fields would fail the first hash test in the previous step also.

6. The transaction is now recorded in the transaction history and will be referenced when polling the balance of either peer (public key) in the future.

Transaction History Hash Chain

How does Timekoin keep a consistent history of transactions? It does this by chaining each transaction cycle to the next via the SHA256 hash that is generated by each transaction. Since all the encrypted data in a transaction is secured with a SHA256 hash, each transaction cycle is secured by making a SHA256 hash of all the other SHA256 hashes in that specific transaction cycle.

A security hash is recorded in the transaction history with a "**H**" attribute. The other fields for the encrypted data are simply padded with a template binary data to fill the space. This transaction record is simply a combined data string of all the previous transactions SHA256 hash which is then built into it's own separate SHA256 hash and stored in the database.

timestamp	public_key_from	public_key_to	crypt_data1	crypt_data2	crypt_data3	hash	attribute
1338580200	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	99aa5949a65d2	H
1338580241	-----BEGIN PUBLIC KEY----- MIHfMA0GCSqG	-----BEGIN PUBLIC KEY----- MIHfMA0GCS	t4BZ9BaS8 D/VhmpN5	PZUnvFpGcG H0d38dja1dX	dQ3KbdLqVIltt j83AdjjAG6M	84919ace88eee	T
1338580244	-----BEGIN PUBLIC KEY----- MIHfMA0GCSqG	-----BEGIN PUBLIC KEY----- MIHfMA0GCS	t4BZ9BaS8 D/VhmpN5	PZUnvFpGcG H0d38dja1dX	dQ3KbdLqVIltt j83AdjjAG6Mf	a3jd19ace88eee	T
1338580252	-----BEGIN PUBLIC KEY----- MIHfMA0GCSqG	-----BEGIN PUBLIC KEY----- MIHfMA0GCS	t4BZ9BaS8 D/VhmpN5	PZUnvFpGcG H0d38dja1dX	dQ3KbdLqVIltt j83AdjjAG6M	j3h8djjz848eee	T
1338580500	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	fh378949a65d2	H

This example above illustrates how the SHA256 hashes are built and chained together.

The first part is the transaction cycle. This encompasses everything from 1338580200 to 1338580499 unix time wise. As you notice, the Hash is in green and the transactions are in blue. When the next hash is being produced, it will take the hash of those 3 transactions and the previous hash and combine them into one long string.

Example would be: **99aa5949a65d2 84919ace88eee a3jd19ace88eee j3h8djjz848eee**

This is then converted into a single SHA256 Hash resulting in the **fh378949a65d2** that is shown in the orange field.

This process continues, stacking more and more SHA256 hash strings from the beginning transaction all the way to the present time as the cycle repeats over and over.

When there are no transactions to build the chain, there will always be a previous hash to work with. So even with no transaction activity taking place, the SHA256 chain can continue forever as per the example below.

timestamp	public_key_from	public_key_to	crypt_data1	crypt_data2	crypt_data3	hash	attribute
1338580500	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	fh378949a65d2	H
1338580800	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	a83jbj8dj44492	H
1338581100	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	dabjd838fgh421	H
1338581400	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	8a82034j89dd3	H
1338581700	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	2djz48gja81h77	H

Transaction Foundations

A Transaction Foundation is a very special SHA256 hash. Instead of rolling all the hash data of a transaction cycle into a single SHA256 hash, over 500 transaction cycles are rolled into a single SHA256 hash. The purpose of this is to allow large blocks of history to be easily compared for any difference. When Timekoin peers are comparing transaction history, it would not be practical to load the entire database into memory to produce a comparison hash as the CPU speed and memory requirements would push Timekoin past any modern desktop system.

To speed up this process, each Transaction Foundation is created every 500 transaction cycles and then these are used to quickly create a transaction history hash for comparison. Transaction Foundations can also be thought of as a way to archive the transaction history because only 100% control of the Timekoin network would be able to change any data in these archived sets or tamper with the transaction foundation produced from them.

The fields highlighted in green below in the example chart represent all the data that goes into each Transaction Foundation. The reason the 3 crypt fields are not included is because these are already locked into place with the existing SHA256 hash for the transaction. Thus tampering with the data would change the hash and no match will take place for the data integrity to pass any checks. The extra CPU and memory to process those fields would not be necessary and thus are left out.

timestamp	public_key_from	public_key_to	crypt_data1	crypt_data2	crypt_data3	hash	attribute
1338580200	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	99aa5949a65d2	H
1338580241	-----BEGIN PUBLIC KEY-----MIHfMA0GCSqG	-----BEGIN PUBLIC KEY-----MIHfMA0GCS	t4BZ9BaS8D/VhmpN5	PZUnvFpGcGH0d38dja1dX	dQ3KbdLqVIlttj83AdjjAG6M	84919ace88eee	T
1338580244	-----BEGIN PUBLIC KEY-----MIHfMA0GCSqG	-----BEGIN PUBLIC KEY-----MIHfMA0GCS	t4BZ9BaS8D/VhmpN5	PZUnvFpGcGH0d38dja1dX	dQ3KbdLqVIlttj83AdjjAG6Mf	a3jd19ace88eee	T
1338580252	-----BEGIN PUBLIC KEY-----MIHfMA0GCSqG	-----BEGIN PUBLIC KEY-----MIHfMA0GCS	t4BZ9BaS8D/VhmpN5	PZUnvFpGcGH0d38dja1dX	dQ3KbdLqVIlttj83AdjjAG6M	j3h8djjz848eee	T
1338580500	11100010111010	11100010111010	1110001011	1110001011101	1110001011101	fh378949a65d2	H

The above data set for the transaction cycles would produce a SHA256 hash like the example below:
6916059b5ee91462ac960387dj8335a07a613a7cd0dha1778cc98e43052c74c0

Summary

This document should contain enough technical information to help make sense of all the source code for Timekoin. The source code should be well documented enough to also aid in understanding how the system works and what would be needed to interface with other peers of the Timekoin network. This is provided that all the protocol standards are followed when communicating with other peers.

Enjoy!
The Timekoin Team

Join the Timekoin Community. Debate and Discuss with us about the future of crypto-currency.
http://timekoin.net/