

A NEW DESIGN FOR A FLEXIBLE INTEGRATED PHASOR
SYSTEM

By

Andrew H. Armenia

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Major Subject: COMPUTER AND SYSTEMS ENGINEERING

Approved:

Joe H. Chow, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

May 2010
(For Graduation May 2010)

CONTENTS

LIST OF TABLES	ii
LIST OF FIGURES	iii
ACKNOWLEDGMENT	iv
ABSTRACT	v
1. INTRODUCTION AND HISTORICAL REVIEW	1
2. NETWORK COMMUNICATION ISSUES IN SYNCHROPHASOR SYSTEMS	3
3. DISTRIBUTED DATA COLLECTION, STORAGE, AND SHARING	6
4. DATABASE SYSTEM DESIGN	8
5. DATA ROUTING	11
6. INTEGRATION OF CRYPTOGRAPHY	13
7. USER INTERFACE AND EXTERNAL APPLICATION INTERFACE DESIGN	18
8. CONCLUSIONS	21
LITERATURE CITED	21

LIST OF TABLES

4.1	Data insertion rates into MySQL tables	8
6.1	Time to execute cryptographic algorithms on 512-byte test message. . .	17

LIST OF FIGURES

1.1	An overview of FIPS	2
2.1	TCP congestion window dynamics	4
4.1	Buffering inbound data to the database	9
5.1	IEEE C37.118 data frame format (simplified overview)	11
5.2	IEEE C37.118 configuration frame format (simplified overview)	12
5.3	SPSP protocol data unit format	12
7.1	Integration of PMU data with Google Maps	19
7.2	A listing of event data sets	20

ACKNOWLEDGMENT

The author acknowledges the support of the RPI Power System Research Consortium Industry Members: AEP, FirstEnergy, ISO New England, New York Independent System Operator, and PJM Interconnection. Furthermore, the author thanks J. Ritchie Carroll and Paul Trachian at the Tennessee Valley Authority, and Prof. Yilu Liu and the FNET project at Virginia Polytechnic Institute and later the University of Tennessee, Knoxville, who have provided assistance in gathering data for testing purposes. Finally, the author acknowledges the work of Luigi Vanfretti in developing synchrophasor applications, and that of Scott Ghiocel on user interfaces for phasor data. Please note that no significance is implied by the order of acknowledgements.

ABSTRACT

In power systems, data collection is an important function to ensure reliable operation of the system. Throughout the evolution of power systems, the accuracy, precision and rate of data collection have been the subjects of improvement. Recently, synchrophasor measurement has proven to be an important step in that evolution. Synchrophasor measurement uses GPS signals to provide accurate measurement of phase angles in a power system spanning a wide area. This data is typically collected at a very high sample rate, such as 30 or 60 samples per second. Transmitting this data to various locations for processing and analysis entails a number of network and communication issues. Furthermore, storage of the large volume of data arising from synchrophasor measurements is an area which has not received significant attention. A new synchrophasor system design, called FIPS (Flexible Integrated Phasor System) has been proposed to address these issues.

It is important to address the limitations of existing communication protocols in transmitting synchrophasors. Many protocols exhibit significant overheads which can result in additional bandwidth costs or delays. Minimizing these overheads is an important part of a synchrophasor system. Also, distributed communication and processing can provide significant improvements to a wide area measurement system. By distributing the data collection and storage, the communication network between systems needs only to transmit the data that is desired by the data consumers, reducing communication costs. By making appropriate use of cryptographic algorithms and techniques, it is shown that data integrity and confidentiality can be guaranteed, even in a distributed peer-to-peer phasor system.

Furthermore, synchrophasor data presents a new set of demands on a data storage system. It is shown that assumptions about the nature of data arrival can be used to construct a database with improved performance characteristics. Additionally, an extensible interface for users and applications to access phasor data stored in a phasor data system has been developed.

1. INTRODUCTION AND HISTORICAL REVIEW

Presently, the deployment of synchrophasor measurement in power systems is rapidly expanding, while applications are less widespread. Existing applications include phasor state estimation [1] and line parameter estimation [2]. In current synchrophasor networks, data is routed to large data concentrators in central locations. Data is then routed from those data concentrators to data consumers using protocols such as the Inter-Control Center Communication Protocol, or ICCP [3].

Due to the centralized nature of these systems, the central points have significant communication and storage demands placed upon them, especially as the number of PMUs increases. This has been observed at the Super PDC installed at the Tennessee Valley Authority [4].

Existing synchrophasor systems do not typically account for the possibility of network delay or data loss in an efficient manner. These systems use existing reliable data transport protocols, such as the Transmission Control Protocol (TCP), to reduce the possibility for data corruption and loss. However, TCP is a poor choice for synchrophasor data, because retransmission delays can significantly impact the latency of a communication channel using TCP.

Existing synchrophasor systems often use database systems which are not optimized for the application of high-speed time-series data storage. The use of general-purpose database systems in these applications typically results in inefficiency in either space or time. A file-based storage scheme optimized for sequential time-series data storage and extraction eliminates these issues.

Finally, building applications to run on existing synchrophasor systems is difficult, because no standard mechanism for data access has been provided. Thus, a goal of a new synchrophasor system design should be to provide easy access to data by applications.

The Flexible Integrated Phasor System, or FIPS, is proposed as a solution to these issues [5]. An overview of the proposed phasor system is shown in Figure 1.1.

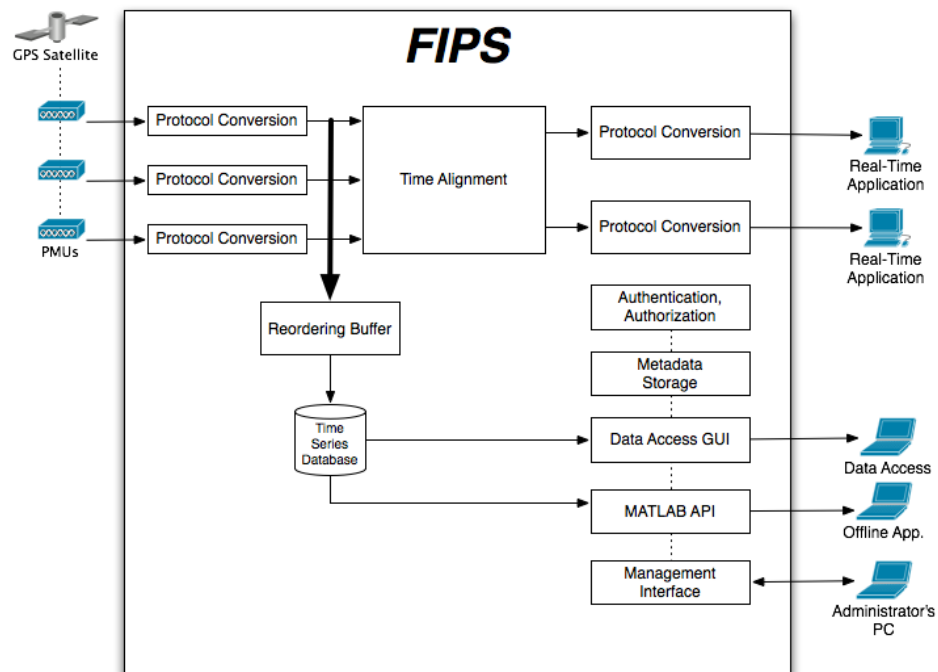


Figure 1.1: An overview of FIPS

2. NETWORK COMMUNICATION ISSUES IN SYNCHROPHASOR SYSTEMS

In a synchrophasor system, the design of the underlying communication network determines the latency and reliability of data transport, among other factors. Latency is an important consideration if the synchrophasor data is to be used for real-time control, because latency in only one communication link results in system-wide delays in the alignment process. Further, it is important that the network design ensures that failures in parts of the system do not lead to widespread outages. In addition, communication protocols should be selected to reduce communication overhead whenever possible. This is because PMUs may be located in remote locations with limited or no access to high-bandwidth communication networks.

Data reliability can be guaranteed to different levels by a number of methods. The IEEE C37.118 protocol [6] for synchrophasor data transport incorporates one of these methods: a cyclic redundancy check. This algorithm checks for bit errors in the received data, but if errors are detected, the data must be discarded. C37.118 makes no provisions for retransmission of corrupted data. C37.118 frames, however, may be encapsulated within another protocol to provide a higher degree of reliability.

The Transmission Control Protocol (TCP) may be used to provide reliable communications for synchrophasor data. TCP, however, implements a congestion window mechanism [7] [8] which is disadvantageous for transmitting constant-bit-rate data. The size of the congestion window determines the amount of data which may be sent before an acknowledgement of previously-sent data is received. The size of this window increases slowly when congestion is not detected, as measured by packet losses or timeouts. When congestion is detected, the window size is rapidly decreased to avoid exacerbating the congestion, as shown in Figure 2.1. To eliminate the possibility of congestion affecting the TCP flows, it is necessary to mark the data packets at the network edge and ensure that sufficient network resources are available to accommodate these flows. Thus, queues within the network will not be overwhelmed by data.

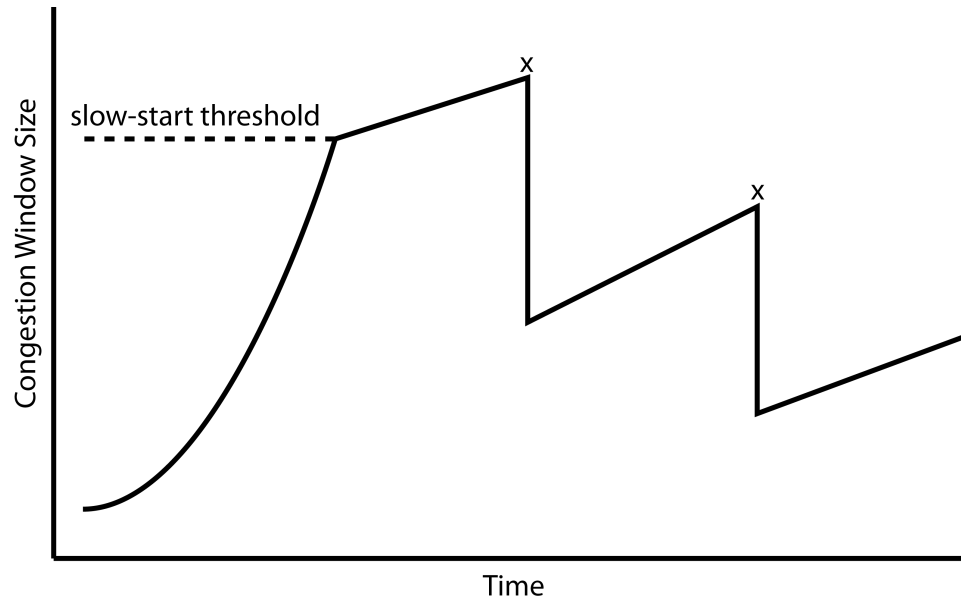


Figure 2.1: TCP congestion window dynamics

TCP, while providing reliable data transport, imposes significant overhead. TCP makes use of complex protocols for congestion management and control, and it has been shown that these mechanisms are in fact detrimental to the transmission of constant bit-rate data. Other protocols may be used to transport the synchrophasor data, such as the User Datagram Protocol (UDP) [9]. UDP provides a simpler protocol for transporting data frames through a network. However, UDP has a frame header of considerable size: it is defined in the protocol as 16 bytes. When combined with a minimum 20-byte IP header, this frame header may represent a significant percentage of total data size, especially in the context of synchrophasor data frames, which may be small. The Realtime Transport Protocol (RTP) provides a solution to this problem. By using RTP's header compression facility [10], the 20-byte IP header and 20-byte TCP header or 16-byte UDP header may be reduced to a 2-byte header. This savings may allow additional synchrophasor data transmission over communication links where it was previously impossible: the 38-byte savings is sufficient to allow the transmission of 9 additional floating-point values per frame.

When using UDP, it is important to note that reliable data transport is no longer ensured by the transport protocol. Thus, the sender and receiver must agree

on a protocol for retransmission of lost synchrophasor data frames. These missing frames can be detected from the embedded timestamps in each frame. If it is assumed that the network packet loss rate is small, and a buffer of recent data can be maintained at the sender, the best choice is a selective-repeat protocol. In this protocol, the receiver and sender do not halt the transmission when errors are detected. Transmission of real-time synchrophasor data continues apace while the error is being corrected. The receiver detects the error and sends a message back to the sender requesting retransmission of the missing or corrupted data. This data can be sent through the network with a lower priority marking to avoid disturbing the existing data transmission. Also, when using a selective-repeat protocol, it is possible to embed more complex retransmission behavior: for example, omitting retransmission of data beyond a certain time threshold.

Redundancy in communication links is also important. No communication protocol alone can provide redundancy against physical damage to optical fiber or copper cables used as communication channels. Redundancy can be provided at the physical layer of the network to account for this possibility. For instance, two optical fibers may be run along separate paths from the source to the destination. The receiver can then be equipped with hardware to discriminate between the two received signals, and automatically select the stronger one. Thus, if one fiber is damaged, communications will not be interrupted, even briefly.

Existing communication networks may make use of the Multiprotocol Label Switching (MPLS) [11] or Frame Relay [12] protocols. It is desirable for the underlying communication network to support multicast traffic, because efficiency of a multi-point data broadcast can be improved by using multicast. MPLS networks support the use of multicast packets [13].

3. DISTRIBUTED DATA COLLECTION, STORAGE, AND SHARING

In synchrophasor systems, it is desirable to have access to data collected over a wide area. However, the large volume of data makes single-point data collection impractical for scaling reasons. First, bandwidth requirements at the single point will grow as the number of data sources increases. Second, storing data at a single point requires large quantities of storage, and could represent a single point of failure. Finally, it is desirable for the collectors of synchrophasor data to be able to share that data with others in certain circumstances. Thus, it is desirable to implement a synchrophasor data collection system in a distributed fashion. Proposals such as NASPInet [14] have also addressed this need.

In a distributed synchrophasor system, data is collected from PMUs at many points. This data is stored locally in a database, and may be combined with data stored remotely in other databases. Furthermore, real-time phasor data streams must be shared between the several locations, to enable wide-area real-time analysis applications.

The distributed database implementation can be done in straightforward fashion, making use of existing Web service technologies to provide an interface to the database. The distribution of streaming data, however, presents a different challenge. It is desired to avoid wasting bandwidth in a situation where one data source is to provide data to a number of consumers. Thus, schemes such as IP multicast should be used to provide a publish/subscribe-type architecture for sharing of data.

In IP multicast, routers and other elements in the network handle copying of data within the network layer. Algorithms [15] [16] are used to compute a minimum spanning tree, shared tree, or other efficient topology for transmitting data to all subscribing nodes. A minimum spanning tree is a tree which connects all nodes in the network with minimum path cost. Due to the copying of data within the network, bandwidth requirements at the network edge are reduced. Furthermore, if desired, IP multicast can be implemented making use of existing point-to-point

links. IP multicast makes use of the concept of group addresses: a group address represents a set of nodes which desire to receive certain broadcasts.

To share synchrophasor data, synchrophasor streams can be mapped onto multicast group addresses. In an efficient system, these addresses are mapped onto indivisible sets of measurements. This is required because, if a measurement is sent to a node where it is not needed or wanted, bandwidth is wasted. Depending upon the network configuration, security mechanisms may be required to ensure that only authorized receivers may read the transmitted data, and only authorized senders may transmit data in the network. This is discussed later in further detail.

4. DATABASE SYSTEM DESIGN

Database systems for synchrophasor applications require certain properties. They must be able to handle a high input data rate, and they must be able to handle large table sizes while sustaining that data rate. When using general-purpose database systems, it may be difficult to ensure these properties, but by constructing a custom data file format leveraging reasonable assumptions on the nature of the data, it is possible to improve the efficiency of the database system.

General-purpose database systems, including MySQL [17], PostgreSQL [18], and Berkeley DB [19] were tested initially for static storage of synchrophasor data. This was done in order to avoid the need to develop a custom database system. When data was loaded into the database tables, it was observed that the index files quickly grew to a size comparable to that of the data files. Indexing is necessary in database systems, because data is not necessarily stored in the data files in a sorted order. However, the index presents an additional overhead which is not necessary if the data is stored in sorted order.

Furthermore, general-purpose database systems were proven to be only partially suitable to the demands of real-time synchrophasor data capture, as shown in Table 4.1. This is due to the observation that table insert rates decrease following a logarithmic trend as table sizes grow. Because many indexing trees use $O(\log n)$ algorithms to balance the index trees upon insertion [20], this behavior is expected. Also, the nature of the input data represents a worst case: the data typically arrives in approximately-sorted order, leading to very unbalanced index trees.

Because the data does arrive in approximately sorted order, a custom data

Number of Rows	Rows / Sec	Number of Channels (at 30sps)	Time to Fill (min)
100,000	65,100	2,170	0.03
1,000,000	56,900	1,897	0.29
10,000,000	49,700	1,655	3.36

Table 4.1: Data insertion rates into MySQL tables

format can be implemented which achieves a high level of time efficiency without wasting space. Incoming data to the database can be buffered for a set period of time. The data in this buffer is maintained in sorted order, and the buffer period compensates for out-of-order arrivals within a certain time period. The time period may be adjusted to provide additional reliability in data capture at the expense of latency, or vice versa. Data from the buffer is written to the data file strictly in order of increasing timestamps. As shown in Figure 4.1, the dark-colored cell must be filled before any of the data points above it in the queue can be written to the database. Alternatively, the red cell may be discarded from the queue. Finally, if all data must be captured, data points arriving grossly out of order, i.e. the discarded red cells, may be captured into a separate file for integration into the database at a later date.

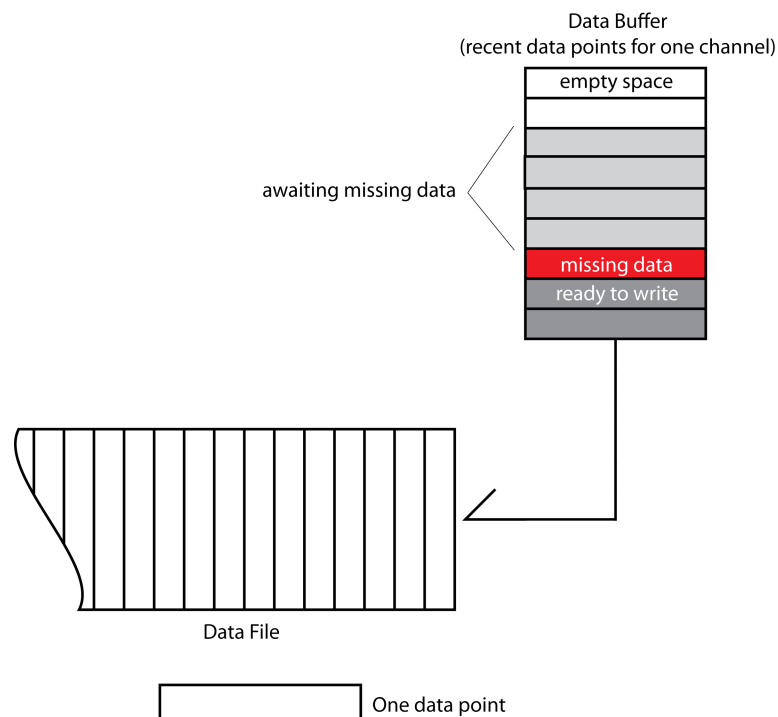


Figure 4.1: Buffering inbound data to the database

Once data has been captured in a strictly ordered fashion, possibly with a few out-of-order points stored elsewhere, search can be conducted efficiently using the well-known binary search algorithm. When using this algorithm, the data set

to be searched is partitioned in half. Then, either the left half or the right half is searched, depending on whether the value sought is greater or less than the value at the partition. This division of the data set continues until either the value sought is found or no more data remains. It can be seen that this algorithm requires at most $\log_2(n)$ divisions of a data set of size n to find an arbitrary timestamp within the data. In this storage scheme, no space is wasted unnecessarily on indexing. The self-indexing provided by storing data in order also has time efficiency on the same order as tree-based indexes.

5. DATA ROUTING

FIPS is composed of several components: networked communication, data acquisition, data storage, and data access. Therefore, it is essential that these components be integrated in a manner which provides the necessary system-level functionality. The overall layout of FIPS is shown in Figure 1.1. Initially, it is noted that the functionality may be divided into a latency-critical path and a non-critical path.

The latency-critical path includes data acquisition and network communication. Data is captured from a number of sources which use diverse formats. Thus, an intermediate protocol, called Simple Phasor Streaming Protocol, or SPSP, is introduced. Then, the data routing engine needs to work only with data in SPSP format. In addition to being a single format, SPSP has the advantage of containing identifying information with each measurement. The IEEE C37.118 protocol includes similar information, but the information is not transmitted with each measurement, so state information must be maintained for each connection. Therefore, processing of SPSP data frames from diverse sources can be done with greater memory efficiency than processing of C37.118 data frames. The differences between SPSP and C37.118 data frames are shown in Figures 5.1, 5.2, and 5.3.

IP Header
TCP Header
Timestamp
Frame Type
Phasor Values
Analog Values
Digital Values

Figure 5.1: IEEE C37.118 data frame format (simplified overview)

Once data has been converted to SPSP format, it can be passed to the different FIPS components. This is the job of the routing engine. This component provides streams satisfying the requirements of different data consumers. For instance, real-time applications such as phasor state estimation should be provided with a time-

IP Header
TCP Header
Timestamp
Frame Type
Phasor Types and Names
Analog Types and Names
Digital Types and Names

Figure 5.2: IEEE C37.118 configuration frame format (simplified overview)

IP Header
UDP Header
Timestamp
Channel ID — Value
...
Channel ID — Value

Figure 5.3: SPSP protocol data unit format

aligned data stream to avoid errors, but raw, unaligned data should be provided to the database to avoid data loss. The routing engine, however, simply must accept data frames from sources and route them to the appropriate destinations. Functionality such as time alignment can be implemented by routing the data to be aligned into the alignment engine. This data is then routed back into the routing engine to be routed to its final destination.

A time alignment engine is also an important component in FIPS. Time alignment involves a tradeoff between latency and reliability. Consider the case in which one PMU occasionally provides a data point which has been delayed by many seconds. It is clearly undesirable to delay timely data waiting for significantly untimely data. But if the data is delayed only briefly, depending on the needs of the application, it may be desirable to wait a brief period for the data to arrive. The maximum allowable delay, therefore, is a tunable parameter of the time alignment engine. When combined with the FIPS routing engine, it is even possible for data alignment engines operating in this way to provide multiple data streams satisfying the requirements of diverse applications.

6. INTEGRATION OF CRYPTOGRAPHY

Security is of critical importance if a solution to the problem of synchrophasor data collection is to be adopted in the power industry. In order to design the necessary security measures to be implemented in FIPS, it is first necessary to evaluate the different threats against the system. Only once the threats have been defined can the proposed security measures be evaluated for their resistance to attack. Also, the severity of each threat must be weighed against costs imposed on the system by the necessary countermeasures. These costs will arise because any cryptographic algorithm intended to ensure a certain property will require time to run. This may add latency to the data pipeline, or require more computing resources to process the data at a sufficient rate.

One threat to be considered is the possibility of data being intercepted and viewed without authorization. A lack of data confidentiality does not present an immediate threat to the safe or reliable operation of the power system. However, some providers of synchrophasor data may not wish for other market participants to view their synchrophasor data, because they view it as commercially sensitive. It is therefore important that an efficient mechanism for ensuring data confidentiality in a multi-participant network be built into FIPS.

A second threat is the possibility of the introduction of false or malicious data into the phasor system. If the phasor data becomes critical to system operations, i.e. it is presented to operators or automated systems empowered to make control decisions, it is crucial that the data can be trusted. That is, it must be provable to a high probability that the data's purported source is in fact its actual source. In addition to the obvious reliability implications, this functionality is important to ensure compliance with guidelines promulgated by the North American Electric Reliability Corporation (NERC) [21].

A third threat is the threat of denial of service. Denial of service refers to the possibility of overwhelming a system or communication channel with large amounts of spurious requests or data. When a denial-of-service attack occurs, the system

becomes so overwhelmed with the spurious requests that it is unable to perform its intended function. While cryptographic algorithms may be used to prevent certain types of denial-of-service attacks targeted at hosts [22], some attacks may only be mitigated by excluding intruders from critical communication links. This form of security is beyond the scope of FIPS.

Data integrity can be ensured using two classes of cryptographic algorithms. The first, a hash function, is used to compute a short, unique, fixed-length code based upon the contents of a message M . This hash can be considered as a function $H(M)$ with a few important properties. First, it should be difficult to compute an arbitrary M' such that $H(M') = x$ where x is known. A hash function lacking this property is said to be vulnerable to a first preimage attack. That is, in a first preimage attack, the attacker is able to construct a message with a known hash value. Second, it should be difficult to compute a message M'' where $H(M'') = H(M)$ where M (and thus $H(M)$) are known. A hash function lacking this property is said to be vulnerable to a second preimage attack. In a second preimage attack, an attacker is able to manipulate a message without affecting its hash value. Clearly, if a hash function is vulnerable to a first preimage attack, it is vulnerable to a second preimage attack. But the inverse is not necessarily true. Because the unique relationship of a message to its image under the hash function may be relied upon to guarantee security properties, such as the authentic origin of a message, both forms of preimage attacks are undesirable.

Many existing functions, such as MD5 [23] and SHA-1 [24], have not been shown to be vulnerable to preimage attacks at the present time. That is, given current limits on computational power, it is likely difficult to compute an inverse of the functions. But a third property which a hash function should possess is resistance to collision attacks. A collision attack implies the ability of an attacker to efficiently compute two messages, M and N , for which $H(M) = H(N)$. MD5, for instance, has been shown to be vulnerable to collision attacks [25]. However, it will be shown that collision resistance is not important to ensuring data integrity, as long as certain conditions are met.

Most collision attacks rely on the so-called birthday paradox: from the per-

spective of any one individual in a room with 30 people, it is unlikely that another person has the same birthday. There is, however, a significant probability that some pair of people in the room has the same birthday. A similar idea applies to hash functions: among a large body of messages M_1, M_2, \dots, M_n , it is unlikely that for any M_i , $H(M_i) = H(M_j)$ for some arbitrary j and fixed i . But when all pairs of messages M_i and M_j are considered, the probability that $H(M_i) = H(M_j)$ for any arbitrary pair becomes significantly larger.

When hashes of randomly-generated keys are used to establish a chain of trust, the birthday paradox leads to a problem. An attacker could generate many messages: some which plausibly originate from the attacker, and others which purport to originate from another party. If enough messages are generated, the birthday paradox means that some pair of messages is likely to have an identical hash. Thus, if a trusted authority indicates that the hash of the legitimate message is valid, the trust is extended to the impersonated message due to reliance on the hash's collision resistance. If the message being trusted in this way is in fact a message that the attacker is to be a trusted party, the illegitimate trust can be extended indefinitely [26].

However, collision attacks are less important for securing synchrophasor data. While an attacker might be able to exploit the birthday paradox to generate pairs of synchrophasor messages with identical hash values, it is unlikely that either half of the pair would occur in a real data stream, or that its counterpart would cause some action useful to the attacker if injected into the system illegitimately.

The second component to ensuring data integrity is an asymmetric-key cipher, such as the well-known RSA cipher [27]. These ciphers can be considered as functions of two arguments, a key and a message. Keys for these functions are formed in pairs (K_d, K_e) with certain properties required by the algorithm. The algorithm itself can be modeled as two functions: $C = E(P, K_e)$, and $P = D(C, K_d)$. It is assumed that knowledge of the internal workings of the E and D functions is known. However, the keys K_e and K_d may be kept secret. This class of algorithms can be applied as follows: for a message M , compute $H(M)$ using a hash function. Then, compute $E(H(M), K_e)$ using a value of K_e which is kept secret, but whose corresponding

value of K_d is known to the receivers. Attach this value to the message to be sent. Then, a receiver of M computes $H(M)$, and compares it with $D(E(H(m), K_e), K_d)$. If the values match, it is unlikely that the message arrived from an unauthorized source.

Data confidentiality can also make use of asymmetric-key ciphers. For instance, data could be sent encrypted with an authorized receiver's K_e . If the receiver keeps K_d secret, no one else can receive messages intended for that receiver. But asymmetric-key ciphers tend to be less efficient than another class of ciphers: symmetric-key ciphers such as Blowfish [28]. These are functions $C = E(P, k)$ and $P = D(C, k)$; both encryption and decryption functions use identical keys. The sender will periodically chooses a random value of k , or a session key. This key may be sent to authorized receivers by encrypting it with their asymmetric keys K_e . The authorized receivers may then decrypt the actual data using k , while unauthorized receivers are unable to compute k because they lack a suitable decryption key which would reveal it. If a previously authorized receiver must be deauthorized, simply broadcasting a new value of k to the still-authorized receivers suffices to remove access. If a fast symmetric-key cipher such as Blowfish is used, additional latency can be minimized.

Performance testing of the cryptographic algorithms described has been carried out on the RPI FIPS prototype server. The implementations of the algorithms found in OpenSSL were used for the testing. Results of the testing are shown in 6.1. All operations except for the RSA signature could be carried out well within the constraints of a 30-sps sample rate. The speed of digital signatures could be increased by using a symmetric-key signature algorithm such as HMAC. This algorithm produces a keyed hash using three invocations of an underlying hash algorithm [29].

Algorithm	Average Wall-Clock Time Per Operation (1000 trials) (ms)
Blowfish Encryption	3.80
MD5 Hash	2.34
SHA-1 Hash	2.35
RSA Signature	23.33

Table 6.1: Time to execute cryptographic algorithms on 512-byte test message.

7. USER INTERFACE AND EXTERNAL APPLICATION INTERFACE DESIGN

A synchrophasor system must provide efficient access to the data captured. It is important that this data be made available in formats suitable for both manual analysis and input into automated analysis applications. To that end, FIPS provides a graphical user interface to permit users to interact with the system and extract data. FIPS also provides an application programming interface accessible from applications developed in a multitude of languages.

The FIPS interfaces are built using the Ruby on Rails web framework [30]. This framework implements a Model-View-Controller architecture, which separates data access and selection from data presentation. Using this architecture, it is straightforward to develop a system which provides access to data through many interfaces and formats.

The FIPS graphical user interface provides several functions to a user of the system. First, the GUI provides a means to store data about PMUs and their constituent measurement channels. This is important because, in the database, data is stored only with a unique channel identifier. Users require this metadata to be able to identify the meaning of a measurement extracted from the database. Furthermore, this data is used to provide the user with a view of PMUs on a map. This functionality makes use of the Google mapping API [31]. An example of this functionality is shown in Figure 7.1.

Second, the FIPS GUI provides a means to track events. These events represent data sets of particular interest. The GUI allows a user to extract data within the timeframe of an event in a variety of formats. This data extraction functionality is extensible to support additional formats as necessary in the future. An example of the FIPS event tracking UI is shown in Figure 7.2.

FIPS also provides data access functionality to automated applications via web service APIs. These APIs mirror the functionality which is exposed to a user via the GUI. Many programming languages support the data formats used by these APIs;



Figure 7.1: Integration of PMU data with Google Maps

to date, an implementation for MATLAB has been developed. For applications requiring real-time data access, the use of web services could cause excessive latency. For these applications, it is recommended to make use of a data stream in the IEEE C37.118 or a similar format.

Event Name	Date	Time	Offset	Year	Actions
Florida Blackout NY	Tue Feb 26	07:46:00	-0500	2008	Show Edit Destroy Get Data
Florida Blackout 1 Day	Tue Feb 26	00:00:00	-0500	2008	Show Edit Destroy Get Data
Superbowl	Tue Feb 26	23:59:00	-0500	2008	Show Edit Destroy Get Data
Testing	Sun Feb 03	18:00:00	-0500	2008	Show Edit Destroy Get Data
Nigeria 21 VT FDR	Sun Feb 03	05:00:00	-0500	2008	Show Edit Destroy Get Data
Nigeria 20 VT FDR	Sun Feb 03	10:00:00	-0500	2008	Show Edit Destroy Get Data
Nigeria 19 VT FDR	Sat Jan 27	12:08:31	-0500	2007	Show Edit Destroy Get Data
Nigeria 18 VT FDR	Sat Jan 27	10:23:41	-0500	2007	Show Edit Destroy Get Data
Nigeria 17 VT FDR	Sat Jan 27	10:23:41	-0500	2007	Show Edit Destroy Get Data
Nigeria 16 VT FDR	Fri Jan 26	19:00:00	-0500	2007	Show Edit Destroy Get Data
Nigeria 15 VT FDR	Fri Jan 26	15:57:51	-0500	2007	Show Edit Destroy Get Data
Nigeria 14 VT FDR	Tue Jan 23	07:00:00	-0500	2007	Show Edit Destroy Get Data
Nigeria 13 VT FDR	Tue Jan 23	15:57:58	-0500	2007	Show Edit Destroy Get Data
Nigeria 12 VT FDR	Mon Jan 22	19:00:00	-0500	2007	Show Edit Destroy Get Data
Nigeria 11 VT FDR	Mon Jan 22	15:57:59	-0500	2007	Show Edit Destroy Get Data
Nigeria 10 VT FDR	Mon Jan 22	17:55:21	-0500	2007	Show Edit Destroy Get Data
Nigeria 9 VT FDR	Sat Dec 02	07:00:00	-0500	2006	Show Edit Destroy Get Data
Nigeria 8 VT FDR	Sat Dec 02	08:10:24	-0500	2006	Show Edit Destroy Get Data
Nigeria 7 VT FDR	Sat Dec 02	02:06:36	-0500	2006	Show Edit Destroy Get Data
Nigeria 6 VT FDR	Sat Dec 02	05:55:21	-0500	2006	Show Edit Destroy Get Data
Nigeria 5 VT FDR	Thu Nov 30	03:54:56	-0500	2006	Show Edit Destroy Get Data
Nigeria 4 VT FDR	Thu Nov 30	05:15:20	-0500	2006	Show Edit Destroy Get Data
Nigeria 3 VT FDR	Wed Nov 29	02:06:05	-0500	2006	Show Edit Destroy Get Data
Nigeria 2 VT FDR	Wed Nov 29	03:13:50	-0500	2006	Show Edit Destroy Get Data
Nigeria 1 VT FDR	Tue Nov 28	12:50:44	-0500	2006	Show Edit Destroy Get Data
Nigeria 0 VT FDR	Tue Nov 28	14:05:20	-0500	2006	Show Edit Destroy Get Data
	Tue Nov 28	02:06:05	-0500	2006	Show Edit Destroy Get Data
	Tue Nov 28	05:14:52	-0500	2006	Show Edit Destroy Get Data
	Tue Nov 28	00:50:47	-0500	2006	Show Edit Destroy Get Data
	Tue Nov 28	00:50:47	-0500	2006	Show Edit Destroy Get Data
	Sun Nov 26	02:04:54	-0500	2006	Show Edit Destroy Get Data
	Sun Nov 26	12:48:29	-0500	2006	Show Edit Destroy Get Data
	Sat Nov 25	06:59:05	-0500	2006	Show Edit Destroy Get Data
	Sat Nov 25	07:06:43	-0500	2006	Show Edit Destroy Get Data
	Tue Oct 17	19:59:05	-0400	2006	Show Edit Destroy Get Data
	Tue Oct 17	19:59:10	-0400	2006	Show Edit Destroy Get Data
	Tue Oct 17	16:31:42	-0400	2006	Show Edit Destroy Get Data
	Tue Oct 17	16:35:13	-0400	2006	Show Edit Destroy Get Data
	Mon Jul 24	07:00:03	-0400	2006	Show Edit Destroy Get Data
	Mon Jul 24	07:27:55	-0400	2006	Show Edit Destroy Get Data
	Mon Jul 03	08:15:01	-0400	2006	Show Edit Destroy Get Data
	Mon Jul 03	09:15:01	-0400	2006	Show Edit Destroy Get Data
	Wed Apr 12	05:10:30	-0400	2006	Show Edit Destroy Get Data
	Wed Apr 12	15:00:01	-0400	2006	Show Edit Destroy Get Data
	Tue Apr 11	20:00:00	-0400	2006	Show Edit Destroy Get Data
	Tue Apr 11	20:00:06	-0400	2006	Show Edit Destroy Get Data

Figure 7.2: A listing of event data sets

8. CONCLUSIONS

Existing phasor systems have provided data collection and streaming services for many years, and more recently have provided centralized data storage services, but have a number of shortcomings which need to be addressed in order for a large-scale phasor system to be deployed. First, network communication protocols must be optimized for use in real-time data transmission to avoid issues arising when generic protocols are used.

Second, a distributed system will be essential to a future synchrophasor measurement system. Centrally collecting the large quantities of data produced by widely deployed PMUs will be infeasible due to communication and storage requirements, and also undesirable as a single point of failure.

Third, a phasor data collection system requires a database optimized for the demands of time-series data collection and extraction. General-purpose database systems exhibit too much overhead to be used in this application. A scheme based on the binary search algorithm exhibits desirable properties for phasor data collection.

Time alignment and data routing are important components in a phasor data system. Time alignment results in a tradeoff between latency and reliability, but this tradeoff may be addressed on a per-application basis. Data routing is essential to the efficient operation of a peer-to-peer phasor data network, and required to tie the system components together.

Given certain network topologies, cryptography is essential to ensure the security of the system. It has been shown that the necessary cryptographic algorithms can be run at sufficient speed on commodity hardware to operate on real-time data flows.

In summary, the feasibility of constructing a networked phasor system, based largely upon commodity hardware and software components, has been demonstrated. The foundation of a flexible user and application interface to the system has been developed. Improvements to be made upon existing systems have also been shown.

LITERATURE CITED

- [1] L. Vanfretti, J. Chow, S. Sarawgi, et al, "A framework for estimation of power systems based on synchronized phasor measurement data," in *IEEE Power and Energy Society General Meeting*, July 2009, pp 1-6.
- [2] D. Shi, D.Tylavski, N. Logic, and K. Koellner, "Identification of short transmission-line parameters from synchrophasor measurements," in *40th North American Power Symposium*, Sept. 2008, pp. 1-8.
- [3] "IEC 60870-6 / TASE.2: ICCP," IEC Standard 60870-6-802, 2005.
- [4] J. Zuo, R. Carroll, P. Trachian, et al, "Development of TVA SuperPDC: Phasor Applications, Tools, and Event Replay," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, July 2008, pp. 1-8.
- [5] A. Armenia and J.H. Chow, "A Flexible Phasor Concentrator Design Leveraging Existing Software Technologies," accepted for publication in *IEEE Transactions on Smart Grid*.
- [6] "IEEE Standard for Synchrophasors for Power Systems," IEEE Standard C37.118, 2006.
- [7] M. Allman, V. Paxson, and W. Stevens, "RFC 2581: TCP Congestion Control," 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2581.txt> - date last accessed 4/19/2010.
- [8] V. Jacobson and M. J. Karels, "Congestion Avoidance and Control," 1988.
- [9] J. Postel, "RFC 768 - User Datagram Protocol," 1980. [Online]. Available: <http://www.ietf.org/rfc/rfc768.txt> - date last accessed 4/19/2010.
- [10] S. Casner and V. Jacobson, "RFC 2508 - Compressing IP/UDP/RTP headers for low-speed serial links," 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2508.txt> - date last accessed 4/19/2010.
- [11] E. Rosen, A. Viswanathan, and R. Callon, "RFC 3031: Multiprotocol Label Switching Architecture," 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3031.txt> - date last accessed 4/19/2010.
- [12] T. Bradley, C. Brown, and A. Malis, "RFC 1490: Multiprotocol Interconnect over Frame Relay," 1993. [Online]. Available: <http://www.faqs.org/rfcs/rfc1490.html> - date last accessed 4/19/2010.

- [13] B. Yang and P. Mohapatra, "Multicasting in MPLS domains," *Comput. Commun*, vol. 27, pp. 162-170, 2003.
- [14] Y. Hu, "Data Bus Technical Specifications for NASPInet," 2008.
- [15] D. Waitzman and C. Partridge, "RFC 1075: Distance Vector Multicast Routing Protocol," 1988. [Online]. Available: <http://tools.ietf.org/rfc/rfc1075.txt> - date last accessed 5/9/2010.
- [16] A. Adams, J. Nicholas, and W. Siadak, "RFC 3973: Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," 2005. [Online]. Available: <http://tools.ietf.org/html/rfc3973> - date last accessed 5/9/2010.
- [17] Sun Microsystems Inc., "MySQL 5.0 Reference Manual," 2010. [Online]. Available: <http://dev.mysql.com/doc/refman/5.0/en/index.html> - date last accessed 5/11/2010.
- [18] PostgreSQL Global Development Group, "PostgreSQL Documentation," 2009. [Online]. Available: <http://www.postgresql.org/docs/> - date last accessed 5/11/2010.
- [19] Oracle Corporation, "Oracle Berkeley DB." [Online]. Available: <http://www.oracle.com/technology/products/berkeley-db/index.html> - date last accessed 5/11/2010.
- [20] L. J. Guibas and R. Sedgewick, "A dichromatic framework for balanced trees," in *Foundations of Computer Science, 1978., 19th Annual Symposium on*, Oct. 1978, pp. 8-21.
- [21] North American Electric Reliability Corporation, "Critical Cyber Asset Identification," 2006. [Online]. Available: <http://www.nerc.com/files/CIP-002-1.pdf> - date last accessed 5/4/2010.
- [22] W. Eddy, "RFC 4987: TCP SYN Flooding Attacks and Common Mitigations," 2007. [Online]. Available: <http://tools.ietf.org/html/rfc4987> - date last accessed 5/4/2010.
- [23] R. Rivest, "RFC 1321 - The MD5 Message-Digest Algorithm," 1992. [Online]. Available: <http://www.faqs.org/rfcs/rfc1321.html> - date last accessed 4/13/2010.
- [24] "Federal Information Processing Standards Publication 180-1: Secure Hash Standard," 1995. [Online]. Available: <http://www.itl.nist.gov/fipspubs/fip180-1.htm> - date last accessed 4/13/2010.

- [25] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128, and RIPEMD," 2004. [Online]. Available: <http://eprint.iacr.org/2004/199.pdf> - date last accessed 4/13/2010.
- [26] M. Stevens, A. Lenstra, and B. de Weger, "Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities," in *Advances in Cryptology - EUROCRYPT 2007*, June 2007, pp. 1-22.
- [27] R. L. Rivest, A. Shamir, and L. M. Adleman, "Cryptographic Communications System and Method," U.S. Patent 4,405,829, September 20, 1983.
- [28] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," in *Fast Software Encryption, Cambridge Security Workshop Proceedings*, December 1993, pp. 191-204.
- [29] H. Krawczyk et al, "RFC 2104 - HMAC: Keyed Hashing for Message Authentication," 1997. [Online]. Available: <http://tools.ietf.org/html/rfc2104> - date last accessed 5/4/2010
- [30] D. H. Hansson, "Ruby on Rails: Documentation," 2010. [Online]. Available: <http://rubyonrails.org/documentation> - date last accessed 5/11/2010.
- [31] Google Inc., "Google Maps API," 2008. [Online]. Available: <http://code.google.com/apis/maps/> - date last accessed 5/11/2010.