CSOUP

Census System for data Organization, Usage, and Protection

6.1800 Design Project Report

Authors: Christopher Wang, Selena Zhang, Vincent Lin

Recitation Instructor: Larry Rudolph

WRAP Instructor: Thomas Pickering

May 9th, 2023

1 Introduction

The government of Fictlandia aims to increase census participation by ensuring Fictlandians only need to fill out one census form per year. Consequently, we designed **the Census System for data Organization, Usage, and Protection**, or **CSOUP**.

CSOUP prioritizes *correctness* and *security*. *Correctness* means that no data is lost or corrupted after entering our system. Complete and accurate data enables the equitable provision of public services like education and healthcare, and it ensures all votes are counted. We monitor correctness at each stage of our system to prevent losses during input, corruption during storage, or drops during transmission. Since correctness is context-dependent, we discuss it throughout the paper.

We define *security* to mean that only authorized parties can access unreleased record data. We enforce two layers of protection: (1) users can obtain only data they are authorized to access, and (2) users can decrypt only data they are authorized to see. Security is critical to respecting Fictlandians' safety, since data leaks could expose the locations of vulnerable populations and marginalized groups. Increasing census participation requires Fictlandians to trust their data is protected. We encrypt all census information before storing it and discuss security in the encryption and key management modules. CSOUP upholds correctness and security at the expense of some performance and storage.

Abbreviations

CSOUP: Census System for data Organization, Usage, and Protection
PMM: physical municipal machines. Each municipality has one PMM for every hundred thousand residents.
VMM: virtual municipal machine. Each VMM is paired with each physical municipal machine and runs on the national cloud.
VDD: virtual distributed database. Each municipality has one VDD distributed across all VMMs for that municipality.
WAL: write-ahead log.
DBMS: database management system.
BP: Bureau of Privacy. An independent set of government officials that checks any data access is compliant with current regulation.

2 System Overview

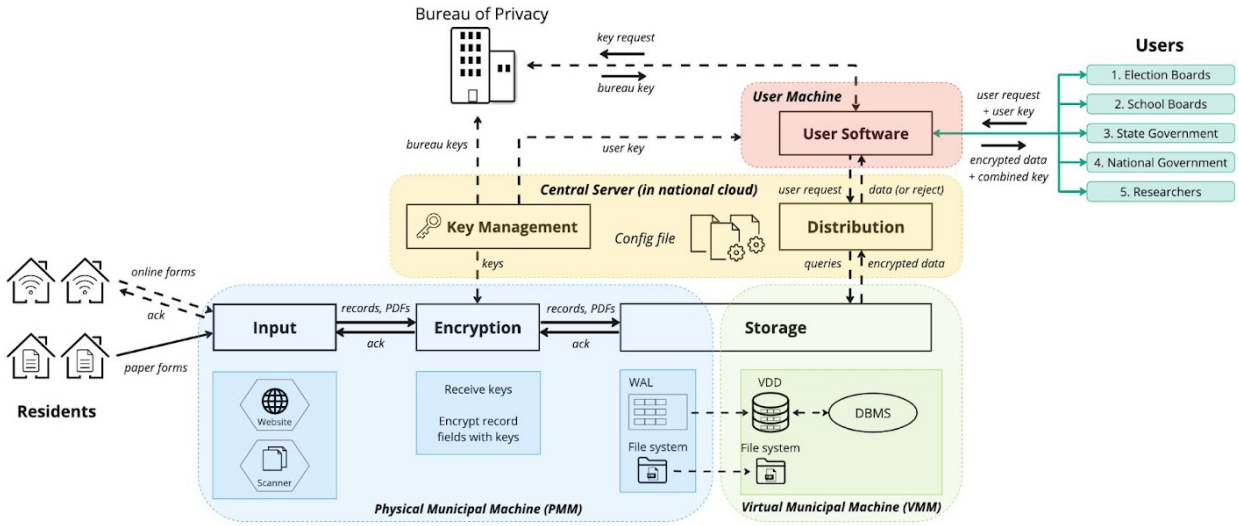

Fig. 1: System diagram of CSOUP showing the main modules, communication between them, and their implementations on machines. Dashed lines indicate network transmission. Acronyms: WAL = write-ahead log, VDD = virtual distributed database, DBMS = database management system. A larger-size diagram is provided at the end of this document.

CSOUP consists of six modules that interact via specified inputs and outputs, each of which can be implemented independently. Below, we describe the **input**, **encryption**, **storage**, and **distribution** modules, as well as the **key management** and **user software** modules that assist their functionality.

The input module (section 3.1) takes in census form submissions, passes them to the encryption module, and acknowledges receipt of the submission once the data has been securely stored. In our implementation, the input module manages network traffic between CSOUP and residents by accepting electronically-submitted census forms, preventing too many users from submitting simultaneously, and monitoring census submission levels to mitigate last-minute submission surges close to the census deadline. Administrative assistants also use scanners to digitize mailed census forms into records and scanned PDFs.

The encryption module (section 3.2) secures records by encrypting different fields of the census records with different sets of keys (depending on which users can access those fields) before passing the records to the storage module. This ordering of modules ensures that no unencrypted data is stored or sent again over the network before being encrypted.

The key management module (section 3.3) enables keys to be correctly distributed to keyholders, and helps create, destroy, and rotate keys. It interacts with two independent government bodies, the

Bureau of Privacy and the Bureau of Key Administration, whose roles are discussed later in more detail.

The storage module (section 3.4) ensures that large quantities of census data are stably stored for the system's lifetime. The storage module is also responsible for receiving data queries from the distribution module and returning the corresponding dataset. At the implementation level, our system achieves this by storing records in triplicate in the VDD, which is managed by a DBMS, and periodically scanning the triplicates to detect and correct any corruption. The DBMS responds to data queries and handles storage balancing across VMMs.

The distribution module (section 3.5) receives pull requests from users and ensures that users receive the data that they are authorized to access. This includes authenticating users, verifying that their requests are valid, and querying the storage module for the corresponding data.

The user software module (section 3.6), which each user installs on a local machine, provides the interface for users to make requests for data. It also takes in the user's key, requests a corresponding key from the Bureau of Privacy, and returns a combined key to the user, who can then use that key to decrypt the data.

A config file, located on a central server hosted in the national cloud, stores values such as the dates for data collection and distribution, frequency of data error checking, the number of keys in use, the key refresh rate, etc. This ensures that changing timelines, frequencies, and keys does not require hardcoding values or changing code; PMMs and VMMs can communicate with the central server to get the values in the config file.

3 System Modules and Implementation

Below, we discuss the implementation of, and interactions between, each of the modules in greater depth.

3.1 Input

The input module is contained within the PMMs in each municipality, which host a web server to handle electronically submitted census forms, which the PMM converts to records. At least one administrative assistant in each municipality also digitizes paper forms using a scanner into records and PDFs. Upon receiving each record and PDF, the input module passes it on to the encryption module. Below, we describe further implementation details for form submission.

3.1.1 Connecting to the Web Server

In periods of low network traffic, Fictlandians (or “clients”) maintain a continuous connection to the web server and complete a web form. The web server saves progress as the user answers each question, preventing them from losing progress if they accidentally quit, their computer crashes, etc.

Each PMM can only support 125 clients submitting the online form in parallel. If too many connect simultaneously, our system supports offline completion of the form and submission when the connection is restored. Too many users submitting at once, however, risks data loss if the PMM cannot receive all the submissions. To maintain *correctness* at the input level, the PMMs in our system limit traffic with a modified version of Explicit Congestion Notification (ECN) with website monitoring. If the number of clients attempting to connect to the server exceeds a preset threshold, then the system will notify, uniformly at random, a number of these clients (in proportion to the excess) that their submission cannot be processed now. By preemptively preventing submissions, the system proactively reduces the risk of data loss.

3.1.2 Network Transmission of Online Submissions

Our system consistently protects against unnoticed data loss with the following network protocol, hereafter referred to as the *whole-data ack protocol*: when a client page submits the online form to the web server in the input module, the input module will respond with an “ack” for the whole submission. Once the client page receives the “ack,” the client knows their submission has been received and can safely disconnect. Otherwise, if the client times out waiting for the “ack,” then their submission failed and they should resubmit. (If the input module’s “ack” gets dropped, the client might resend data that the input module already received. To handle this, the input module keeps track of recently received submissions and discards any duplicates.) We apply this whole-data ack protocol on top of TCP to ensure that both individual packets (TCP) and whole data submissions (our protocol) are transmitted correctly and completely.

One problem is that if the input module acknowledges a submission immediately upon receipt, then a PMM crash could cause the submission data to be lost before it has been written to disk. The data would then be lost from the system, but the client would continue to believe that the submission was successfully received. To handle this scenario, we also enforce correctness using a technique we term as a *delayed multi-ack chain*: When the input module receives a submission, instead of “ack”ing it immediately, it first forwards the data to the encryption module. The encryption module encrypts and sends the data to the storage module, waits for an “ack” from the storage module, and then sends an “ack” back to the input module now that the data has been successfully written to disk. At this point, the data will not be lost even if the PMM crashes. After receiving this “ack”, the input module sends back its own “ack” packet to the web server acknowledging the form submission.

The chain of “ack”s is shown in the diagram in Fig. 1. This delayed multi-ack chain technique comes at the cost of latency, since the client has to wait longer for a response on whether their submission was received, but upholds *correctness*: The ack chain ensures that if such a PMM crash occurs, the client will not receive an “ack” and will attempt to resubmit until the data is securely written to disk.

3.1.3 Monitoring Participation

If too many residents wait until the end of February to submit the census, a traffic surge just before the collection period closes could overwhelm system bandwidth. Dropping submissions during the surge would significantly impair the completeness of the data.

To mitigate this, the input module tracks submissions received throughout January and February and compares participation to the estimated number of residents in the municipality (which can be estimated from last year’s data, or specified in the config file). On checkpointed dates, the system will notify the municipal government if the number of records received is too low. If many municipalities within each state fall below the reasonable threshold, the state and/or national governments will also be notified. This allows the governments to take action to raise participation rates by, for example, increasing advertising and door-to-door campaigns. The config file specifies the checkpoint dates and expected participation numbers.

3.2 Encryption

To ensure *security*, we prevent users from decrypting any data they are not authorized to access, even if they have mistakenly obtained it. To achieve this goal, CSOUP encrypts all census records except for three parts: field headers, high-level location information, and timestamp of submission. Examples of field headers include “name,” “date of birth,” “gender,” etc., but all values associated with each header are encrypted. High-level location information includes the state and municipality in which each record was submitted. These fields are left unencrypted so that the data is searchable and partitionable in aggregate, allowing the appropriate data to be distributed to each user (e.g. a state government should only receive data from its own state). The encryption module receives keys from the central key management module via a secure network connection, ensuring that the records are encrypted via the same keys that users can then later decrypt it with.

Our encryption scheme also protects against error cases where users receive more information per record than they have permission to view. For example, the national government requires a strict subset of the information given to the state government. If the national government administrator obtained information that is only needed by states and municipalities, they should not be able to decrypt those fields. To enforce this layer of security, our system encrypts different sets of record fields with different keys: fields accessible by the national government (e.g name, birthdate, gender) would be encrypted with one key, but licensed car ownership and income supplement eligibility – which are state-only – would be encrypted with another. Similarly, municipal-only fields that are not required by

states are encrypted with yet another key, as are municipal-only fields that are not required by school boards.

Notably, encrypting records trades off with storage space and performance, since it triples the size of each record. A limitation of our system's security is that it does not encrypt PDFs; it is not possible for our system to do so while still meeting the national government's storage requirements, as discussed in section 4.3.

The encryption module forwards all PDFs and encrypted records to the storage module, which responds with an "ack" for each PDF and record once it has been written to disk. This allows the encryption module to "ack" the input module in return, as described in section 3.1.2.

3.3 Key Management

CSOUP ensures that only parties authorized to decrypt records can decrypt them, which requires distributing the correct keys to the correct parties. The key management module coordinates the creation, distribution, and rotation of keys.

We aim to prevent any one government agency from having unilateral access to view data, which degrades regulatory oversight and creates a single point of failure in the case of a malicious attack or information leakage. We also aim to minimize the risk that a complete encryption key is leaked during network transmission, which would compromise security.

To achieve these goals, we propose that a group of national government officials , hereafter termed the Bureau of Privacy (BP), oversee key access and distribution; these officials should be independent of governmental users of census data. CSOUP uses key splitting for all census information: the BP has half of the key for any information, and authorized users have the other. Section 3.6 discusses how users obtain the combined key that allows them to decrypt the data.

Incorporating this key splitting process with the BP enhances security in three ways: (1) It prevents unilateral access to data. For any user to decrypt data, both our system and an independent government bureau must independently approve that user's access to data by each sharing one key. (2) It centralizes enforcement so that changes in access can be enforced immediately. If any data access rights are revoked, it may take time for all users to be notified and change their systems appropriately. The BP, however, can be notified immediately, and if the newly deauthorized user attempts to request data, the BP will simply not provide them with a key. (3) It reduces the risk of distribution because even if an adversary obtains one key, they cannot decrypt the data without both keys.

Introducing the BP, however, creates a major performance drawback: It will take longer for any user to access their census data, because each user must wait for the BP's approval before they can decrypt records, and any delays within the BP will also delay the users. One risk is that the BP's approval process may bottleneck data access during the distribution period: If many users happen to request information on the same day, some will inevitably have to wait on the BP to verify and approve the others. Regardless, we implement this design decision because we prioritize *security* over performance, as leakage of records to unauthorized users could cause Fictlandians to fear that participating could endanger their safety or privacy. Slow performance can be mitigated by the surge protection measures implemented in other stages of the pipeline (for instance, reminders for timely requests sent to users at the distribution level).

A top-level ROLE account can distribute, add, and delete keys. Upon the dismissal of an employee, the ROLE administrator rotates any keys accessible by their role. Since building a foolproof system is impossible, we keep an audit lot to help identify any malfunctions in key distribution. Any time a key is distributed, rotated, or changed in any way, the change is automatically written to the audit log and cannot be deleted.

3.4 Storage

The storage module receives PDFs and encrypted records from the encryption module, and is responsible for persistently storing them and responding to data queries from the distribution module. Below, we detail the implementations for each of these functions.

3.4.1 Census Records

To increase our system's availability in responding to data queries, the storage module stores encrypted records in the virtual distributed database (VDD) shared by each municpality's VMMs.

We do not store records on the PMMs because we prioritize availability over latency. Since the VMMs are highly fault-tolerant and distributed, they provide higher availability for accessing records and greater network bandwidth. Since the VMMs are geographically farther, however, VMM-only storage increases latency for municipalities to access data. We prioritize availability because it is important to meet distribution timelines. Data access, however, is not latency-sensitive.

Given the latency of database inserts (300ms for encrypted records), one potential risk is data loss resulting from a PMM crash between the storage module receiving a record and inserting it in the database. To avoid this, each PMM uses a write-ahead log to log each record received from the encryption module. Once the record is logged, an "ack" is returned to the encryption module to confirm that the data has been written to disk; this "ack" is forwarded back to the user to confirm their submission. Once a VMM receives and inserts the record into the VDD, it informs the PMM to then commit the transaction.

The write-ahead log enforces *correctness*: if the PMM or the network fails before a record is installed in the VDD, then upon system recovery, the PMM will reference the log and determine that this change was never committed. Thus, it will re-attempt to send it to the VMM until the record is successfully installed, ensuring that no individual record is lost.

To additionally uphold the correctness of PMM-VMM transmission, we apply the same *whole-data ack protocol* from section 3.1.2: on top of TCP, the PMM waits for an "ack" from the VMM for each record, and resubmits if it does not receive one.

We allocate a new table for each year's census records, where the table columns consist of the fields for each record. Since each record is encrypted at the field level and not at the full record level, each record's (encrypted) data may be stored and sorted by all fields of the census. This ensures security within the storage module while maintaining organization within the VDD.

3.4.2 Census PDFs

PDFs are persistently stored in PMM file systems. Each PDF is assigned a UUID to identify it in the file system, and each digital record created from the PDF stores this UUID, as well as a unique identifier for the machine on which it's stored, for future reference. CSOUP initially stores all PDFs in PMM file systems, but begins moving PDFs by oldest timestamp into cloud storage at 80% capacity to mitigate the scaling bottleneck posed by the PDFs' large file size. Assessment of this bottleneck is described further in section 4.3.

3.4.3 Database Management System

The VDD within each municipality is coordinated by a database management system (DBMS) for storage and data retrieval functionality. The DBMS storage responsibilities include the storage correctness functionalities in section 3.4.4. The DBMS data retrieval responsibilities include receiving and processing distribution requests and parallelizing VMMs when sending data by delegating up to 100,000 records from the VDD to each VMM for distribution, allowing for distribution to scale with municipality size. The mechanics of communication with the distribution module for data retrieval are described in section 3.5.

3.4.4 Correctness

One major point of concern with any long-lasting system is the integrity of data while idle in storage. Given that our system must store records for 70 years before being publicly released (and afterward as well), data degradation is a substantial threat to the correctness of persistent data. All previously collected census data must be regularly maintained and ready to distribute accurately at all times to accommodate users such as researchers, who may request data at any time, as well as users during the normal distribution period.

CSOUP enforces correctness in the storage module by triplicating records in the VMMs as a means of both detection and recovery of compromised data integrity. This trades off with storage space because three times the storage is required for each record. Ultimately, however, this cost is not as significant given the storage bottleneck posed by PDFs, which occupy orders of magnitude more space than records (section 4.3).

Detection
To detect data degradation within the VDD, the DBMS executes monthly parity checksumming. To ensure the correctness and availability of data, data integrity must be monitored throughout the year as well as just before the distribution period to mitigate any long-term effects of data degradation. The frequency of checksumming, local time of checksumming, etc. are specified in the config file.

Recovery
After detecting a corrupted record, the DBMS obtains its triplicates and restores the majority value. To locate the other two copies, each record stores the unique identifier of the other two copies. This allows the DBMS to quickly access duplicates and resolve data corruption efficiently.

3.5. Distribution

The distribution module, implemented on a central server in the national cloud, allows users to request census data via a software module they install on their machine, and is responsible for ensuring that users obtain the correct census data. Upon receiving a user request, the distribution model authenticates their credentials to determine whether they are authorized to request data; if not, it rejects the request and notifies the governments and system engineers that an unauthorized user attempted to request data.

Upon receiving an authorized user request, the server checks what subsets of data the user is allowed to access, which is determined by the census specification and listed in the config file. If the request is within the bounds of what the user is allowed to access, then the server queries the appropriate VMMs in the storage module for the corresponding data.

To locate the VMMs relevant to the request, a two-level DNS-style tree structure will resolve which municipalities to contact. Each state in Fictlandia will have a designated “nameserver” machine (whose addresses are stored in the central server) that contains the DBMS address of each of that state’s municipalities. The VMM designated to each state is specified within the central config file, but can be arbitrarily assigned, as the additional workload is low and only involves relaying TCP communication between the central server and the state’s municipalities.

Since database reads take a non-negligible amount of time, and potentially many or all municipalities may be contacted to fulfill a request, the server completes the distribution in two stages to reduce the risk of data loss or network failure, promoting correct and complete data transmission. In the first stage, the server notifies the VMMs in all relevant municipalities of the requests, and an initial preparation “ack” must be received from each municipality to confirm that the request was successfully received and data is ready to be sent. In the second stage, the server receives the data from the VMMs and forwards them to the user. At all points in communication (between VMMs, server, and user), we continue to employ the whole-data ack protocol to ensure correct and complete data transmission.

3.6 User Software

CSOUP’s users request data via the user software interface. Although installing software on the user’s machine introduces potential setup complications, we believe this decision is justified because the installed software allows for data decryption to happen at the application level while still keeping the BP key hidden from the user (section 3.3). Local decryption removes any threat of network-based attacks intercepting unencrypted data, and hiding the key ensures that CSOUP and the BP are two independent layers of security to protect sensitive census data.

Accessing data requires a user to perform two requests: (1) send a key request to the BP, who must approve the ticket by sending back a key, and (2) send a request to CSOUP’s distribution module for data. Data will be sent to the user software as soon as the request is received, but the user cannot decrypt the data unless the software module also receives the key from the bureau. To initiate the BP request, the user must enter their own key distributed to them by the key management module. Afterward, once the BP fulfills the request and returns the second key, the software module will combine the two keys with a one-way algorithm to produce the final key that the user can use to decrypt the records. This hides the bureau’s key from the user, enforcing separation of keys. Additionally, transmitting two split keys across the network enforces *security* by ensuring that if a malicious agent breaches the network connection and gets one key, they still cannot use it to decrypt the data.

This extra layer of protection enforces security in data distribution, at the cost of potentially additional processing time. Although requiring an additional body to review requests is more time consuming than if users could access the data directly, it ensures that users are authorized to receive and decrypt exactly the amount of data they are predetermined to be distributed.

Throughout the distribution period, CSOUP will periodically send reminders to users to request census data proportional to the percent of users that have yet to do so (adjustable in the central config file). An internal request deadline of three days prior to the census distribution deadline will also be enforced with the reminders to promote timely completion of ticket approval and data distribution.

4. Evaluation

In this section, we define and analyze three evaluation metrics for our system, based on the timeline and storage requirements specified by the national government: (1) the rate at which our system can handle online form submissions, (2) the time to distribute data to users, and (3) the storage lifetime. Additional evaluations of paper submission scanning and PMM-VMM data uploading are included in the appendix.

4.1. Rate of receiving form submissions

Since each PMM can only handle up to 125 users submitting the online form concurrently, one concern is whether all residents can successfully submit their forms online. This is particularly true if many residents attempt to submit simultaneously at the end of the data collection period, which could result in some residents not being able to submit in time. We evaluate this as follows:

Since municipalities have at least one PMM-VMM pair for every 100,000 people, then assuming that load is evenly balanced across PMMs, each PMM is responsible for up to 100,000 submissions during the data collection period. Since one person submits per household, each household has 2.6 residents on average, and 80% of households submit online, the expected number of online submissions per PMM (if everyone participates) is:

$$100{,}000 \text{ residents} / (2.6 \text{ residents/submission}) * 0.8 = 30{,}800 \text{ submissions}$$

Although residents will take 31 minutes on average to complete a form, in periods of high traffic, they can complete the form offline and connect only to download or submit (section 3.1.1). Let us assume, conservatively, that a user who completes the form offline needs to connect to the server for 2 minutes to download and submit the form. Then, the number of submissions a PMM can handle per hour is:

$$125 \text{ submissions} * (1 \text{ submission}/2 \text{ min}) * (60 \text{ min}/1 \text{ hr}) = 3{,}750 \text{ submissions}$$

This number is greater than 10% of total submissions, meaning our system can handle more than 10% of all participants submitting within the same hour. Since the data collection period is spread out over two months, we consider this case an incredibly unlikely worst-case scenario, which our system is able to handle. Moreover, our system also protects against this worst case by monitoring and flagging low participation levels (section 3.1.3), which allows governments to promote participation to reduce the probability of last-minute surges.

Our system can still handle all online submissions even if a municipality experiences a sudden large population increase from one year to the next, without enough notice to add more PMMs. Suppose that a municipal population doubles from one year to the next due to migration, which we consider

extraordinarily unusual for any municipality with more than 100,000 people. Then, each PMM might have to handle up to 200,000 submissions. From the analysis above, each PMM could still handle more than 5% of all participants submitting online within the same hour, and could in theory process all submissions if everyone submitted within the same 20 hours. Thus, our system can easily handle all online submissions over the course of two months.

4.2. Time to distribute data to users

The DBMS for each VDD allows each VMM to serve requests from multiple users in parallel, provided they are not accessing the same records. Additionally, the many VMMs in the storage module can parallelize the work of extracting and returning data in response to data queries.

Consider a query that requires extracting every record collected in a given year, e.g. a query collected by the national government, or from an external researcher looking for aggregate statistics for the entire national population. Since we have one VMM per 100,000 residents, each VMM can read and return 100,000 records to fulfill the query. Since reading an encrypted record takes 210 ms, this process takes:

$$0.210 \text{ s} * (1 \text{ hr}/3{,}600 \text{ s}) * 100{,}000 \text{ reads} = 5.8 \text{ hrs}$$

After our system extracts the corresponding data, it needs to send it over the network to the user. If a query requires the complete set of records nationwide for a given year, then the total size of the data is:

$$(2{,}400 \text{ bytes/record}) * (300 * 10^6 \text{ records}) * (1 \text{ GB}/10^9 \text{ bytes}) = 720 \text{ GB}$$

With a 10-GBps bandwidth out of each cloud machine, it would thus take 720/10 = 72 seconds to send the data from the cloud to a user. This is negligible compared to the 5.8 hours required to extract all the records from the database. That said, if the user's bandwidth is less than 10 GBps, our system will send data at a slower rate to match. If the user's bandwidth is at least 50 MBps, which is a reasonable requirement for the national government or a data-intensive research institute, then it will take 4 hours to send the data over, which is very reasonably short given the one-month timeframe.

We additionally note that most users (except possibly researchers, whose needs are less time-sensitive) will request significantly less than 720 GB of data: national governments only require a subset of record fields, and state/municipal governments and local boards only require data from a particular location. Thus, we can serve all governments' and boards' queries on the order of a few hours, if not less, which easily meets the April 1st deadline as long as users query at least a week in advance (to avoid potential network failures). This is true even if a municipality's population size were to be twice as large as expected in a given year. We strongly encourage all users to request the data as

early in March as possible to avoid delays caused by network failures or competing queries for the same data; our system cannot guarantee delivery by April 1st for requests made within the last few days of March.

To access the records, users also need to request and receive a key from the BP (section 3.3.1), which is bottlenecked by human response time. Given the one-month timeline, we estimate that a 1-2 week response time from the BP would be reasonable, providing users with 2-3 weeks of buffer time in case of system failures, network failures, requesting the key a week late, etc.

4.3. Storage lifetime

The government requires that all data is stored in perpetuity. Although our system cannot store everything forever due to data accumulating each year, we estimate our system's storage lifetime below.

The size of an encrypted record is 200 words * (4 bytes/word) * 3 (encryption) = 2,400 bytes. With one PMM-VMM pair per 100,000 residents, the total size of all records per PMM-VMM pair each year is:

$$(100{,}000 \text{ records/yr}) * (2{,}400 \text{ bytes/record}) * (1 \text{ MB}/10^6 \text{ bytes}) = 240 \text{ MB}$$

The government also requires that we store PDFs. Assuming that 20% of submissions are paper forms, the average PDF size is 2 MB per household, and there are 2.6 people per household on average, the estimated size of all (unencrypted) PDFs per PMM-VMM pair each year is:

$$(100{,}000 \text{ residents/PMM}) * 0.2 * (1 \text{ household}/2.6 \text{ residents}) = 7{,}700 \text{ households}$$
$$7{,}692 \text{ households} * (2 \text{ MB/household}) = 15{,}400 \text{ MB}$$

Then, the estimated time it would take for the records and PDFs to fill up each both 1 TB disks of each PMM-VMM pair is:

$$2 \text{ TB} * (10^6 \text{ MB/TB}) * 1 \text{ year}/(240 + 15{,}400 \text{ MB}) = \textbf{128 years}$$

Ideally, our system would be able to store data for hundreds, if not thousands, of years; however, even while utilizing all storage resources and keeping PDFs unencrypted, we only achieve a lifetime of 128 years. If the national government would like for our system to store all data, including PDFs, for hundreds of years, it will need to provide us with more storage.

This evaluation highlights a tradeoff between security and storage: Our system's storage lifetime is bottlenecked by PDFs, which occupy nearly 200 times more storage per year than records (when both are unencrypted). Thus, although encrypting records minimally impacts our system's lifetime,

encrypting PDFs would reduce the lifetime by a factor of 3, to 43 years, which does not even meet the 70-year mark. Thus, although security is one of our design priorities, we consider this an unavoidable sacrifice to meet the storage requirements. We recommend that the national government consider providing us with more storage so that our system can provide longer-term storage and security for *all* data, including PDFs.

5 Use Cases

Governments, school boards, and election boards require census data at regular intervals to provide their services. External researchers also require census data for studies. We detail below how our system handles each use case.

5.1 National Redistricting

The national government requires census data to determine national legislature districts, so correctness is critical to fairly allocating national representatives. The national government queries the CSOUP user interface every decade, during March. After authenticating the national government official, our distribution module queries the storage module for the national record fields across all municipalities. We use the whole-data ack protocol to ensure data is distributed completely and correctly. Section 4.1.3 shows our system easily meets the April 1st deadline, even with network failures, as long as the national government queries at least a week in advance. We encourage the national government to request data early, given the large size of their dataset.

5.2 Municipal Election Boards

Municipal election boards use census data to determine voter rolls and mail out ballots, so correctness is key to respecting the right to vote. Municipal elections are the most time-sensitive use case, especially for special elections held shortly after the data collection period. We accommodate municipalities by allowing them one high-priority request per month, which our system will serve before other pending queries. Since each municipal dataset is disjoint, our system can serve requests from many municipalities in parallel, enabling distribution within the one-month timeframe.

5.3 School Boards and Student Assignments

School boards are among the system's most data-sensitive users, so correctness is critical: the loss of a single child's record could deprive them of adequate and accessible education. School boards may also want to compare census data across years, i.e. compare this year's and last year's data to identify school-age children who recently moved in. We thus allow users to request data from past years, which can be served in parallel with requests for the current year's data.

5.4 External Researchers

External research positively informs and evaluates policy. To accommodate external researchers, CSOUP accepts queries year-round and can serve disjoint dataset requests from multiple researchers in parallel, returning statistics in aggregate. In March, however, we de-prioritize researcher data requests to avoid blocking more urgent government accesses. We further restrict research queries to 10TB per month (17-minute send time) to prevent groups from hogging system bandwidth. Since researchers are more likely to study subsets of data rather than full records from every resident nationwide, we find this limit reasonable.

6. Conclusion

CSOUP receives, stores, and distributes Fictlandia's census data with *correctness* and *security* as design priorities. CSOUP upholds correctness, i.e. all data is preserved and transmitted without loss or corruption, with a conservative ack-chain input protocol, write-ahead logging, triplication, and frequent checksumming to protect against machine and network failures and data corruption. CSOUP upholds security, i.e. unauthorized parties are prevented from accessing unreleased census data, via encryption of records, key splitting, and the Bureau of Privacy, which prevent users or malicious agents from having unilateral access to data. CSOUP's limitations and unresolved questions include:

(1) the resulting tradeoffs to latency and user timelines, particularly with the introduction of the Bureau of Privacy
(2) the complexity of implementing governmental infrastructure to support the encryption and key management mechanisms
(3) the implementation of balancing online submission load across PMMs, which our report does not address, and
(4) the inability of the system to store PDFs long-term (which also comes at the cost of PDF encryption).

We recommend that the national government, census data users, and other systems engineers consider these issues further before implementing our system.

7. Author Contributions

The authors collaborated on conceptualizing and designing all components of the system. In conducting further research and writing text in the final report, Selena focused on the input, encryption, and key management modules, Vincent focused on the storage, distribution, and user software modules, and Chris focused on the evaluation and use cases. Vincent and Chris created the system diagram; Selena did the final editing pass.

8. Acknowledgments

We would like to thank Larry Rudolph for giving us incredibly detailed and invaluable technical feedback and ideas for our design (one might even call it accurate and complete, maintaining *correctness*), Thomas Pickering for giving us thoughtful feedback and inspiring encouragement on communicating our design, Shuli Jones for aiding us with technical understanding of systems concepts, and Katrina LaCurts and the rest of the 6.1800 staff for teaching this course.

---

9. Appendix

Included below are supplementary evaluations not included in section 4, namely the time to scan mailed paper forms and to upload records from the PMM to the VMM.

9.1. Evaluation: Scanning paper forms

Each municipality has a single scanner that scans 70 pages per minute. Assuming 20% of households submit via paper and each form is 2 pages on average, the expected number of pages per 100,000 residents is:

$$100{,}000 \text{ residents} / (2.6 \text{ residents/submission}) * 0.2 = 7{,}700 \text{ submissions}$$
$$7{,}700 \text{ submissions} * 2 \text{ pages/submission} = 15{,}400 \text{ pages}$$

Thus, for a municipality with 100,000 residents, it takes (15,400 pages) / (70 pages/min) = 220 min = 3.7 hours to scan all pages, which is easily doable by a single administrative assistant. For the largest municipalities with 1,500,000 residents, it would take 15 times as long, or 3.7 * 15 = 55 hours to scan all pages. Although this is a long duration, we recommend that administrative assistants continually scan paper forms as they arrive, such that all forms are scanned within 1-2 days of the end of the period. (Even for smaller municipalities, scanning forms as they arrive stores the data securely in the system in case the forms are lost or damaged.)

9.2. Evaluation: Time to upload records to cloud

During the data collection period, the PMMs continuously upload encrypted records to the VMMs (section 3.4.1). Below, we evaluate the rate of this process.

Since the bandwidths are high (1 Gbps out of PMM, 10 Gbps into cloud), this step is bottlenecked by insertion time into the VDD (300 ms for encrypted records). Thus, a PMM can transfer data to the VMM at a rate of:

$$1 \text{ record} / 0.3 \text{ s} = (3.3 \text{ records/sec}) * (3{,}600 \text{ hr/sec}) = 12{,}000 \text{ records/hr}$$

Since each PMM will process about 100,000 residents' data, received throughout the two-month period, each PMM can easily upload records to the VMM at a rate faster than they are received. If a PMM fails at some point, or a PMM has to receive additional records when another PMM fails, the PMM will not face issues uploading the additional records: Even in an unrealistic worst case where a PMM received all 100,000 records on the last day of data collection, it would only take about 8 hours to upload them to the VMM; the data would all be in the VDD by the next day.