

Securing Our Data Storage Infrastructures

Bob Mungamuru
Stanford University
bobji@stanford.edu

Hector Garcia-Molina
Stanford University
hector@cs.stanford.edu

Abstract—The threats faced by data storage infrastructures can be broadly categorized into two classes – *break-ins* and *data loss*. Unfortunately, defenses against break-ins tend to increase the risk of data loss, and vice versa. We introduce *configurations* as a model for quantifying and managing the break-in and data loss risks faced by a secure data storage system. Configurations can also be used in making technology investment decisions within the Gordon-Loeb framework.

I. INTRODUCTION

Data storage systems are a critical part of our information infrastructure. Disturbingly, the news these days is filled with stories of how storage systems fail – student records being stolen, patient records being misplaced, a photo agency being sued because they lost an artist’s digital images. Failures of our data storage infrastructures are relatively costly, compared to say, a failure of our communication systems. While the latter typically causes a transient (but annoying) service disruption, the former often results in either permanent leakage or loss of critical data. Thus, our strategy for defending our critical information infrastructures should arguably begin with our data storage infrastructures, and move on “outwards” to our networks.

Broadly speaking, we must safeguard our data storage infrastructures against two basic types of failures. Firstly, we must prevent *break-ins* to our system. By “break-ins” we mean attacks on data privacy and confidentiality, or any other event that leads to an unauthorized party reading our sensitive data. Secondly, we must guard against *data loss* – events that cause our own data to become corrupted or unavailable to us.

A key observation is this: defenses against break-ins tend to increase the risk of data loss, whereas defenses against data loss tend to increase the risk of break-ins. Consider, as an extreme example, System *A* which simply deletes all the data that is input to it. From the privacy point of view, System *A* is perfect – there is no chance that the data will ever be leaked or stolen. However, since our data is not preserved, it is not a useful system. At the other extreme, consider System *B*

which proliferates many replicas of its input data across several public sites on the Internet. Replication is clearly good for the longevity of the data. However, System *B* is weak in terms of privacy since anybody with access to the Internet can read our private data.

It is our position that security research has tended to neglect this fundamental tradeoff between data privacy and data longevity. While we have seen tremendous progress in techniques to separately ensure data privacy (e.g., [2], [8]) and longevity (e.g., [7], [9]), there is typically emphasis on just one objective or the other, without examining the intimate relationship between the two ([10] is one of the few exceptions). To fully understand the risks faced by our data storage infrastructures, we must simultaneously assess both the risk of break-ins and the risk of data loss. We must then choose a defensive strategy that jointly mitigates both risks, judiciously balancing between privacy and longevity as our resources and risk tolerances best allow.

In the remainder of this paper, we briefly describe an analysis framework we have developed (as part of the PORTIA project [3]) to assess and mitigate the risks faced by data storage systems. We then discuss how our framework can be used to make decisions regarding investment in information security.

II. MANAGING RISK

We first present a simple model for secure data storage systems, and show how to quantify the risks faced by such systems. We then discuss how we might optimally balance and mitigate such risk. Our model is developed fully in [5] and [6].

A. Configurations

We begin by defining a pair of data operators that will help us capture the tradeoff between privacy and longevity. A *Copy* operator outputs n replicas of its input data, making loss of the input data less likely. A *Split* operator breaks its input into n shares such that all n shares are necessary to reconstruct the input, making break-ins less likely.

Copy and Split operators are abstractions of real-world operations that are used to safeguard data. For example, a Copy can represent a simple tape backup scheme, or a complex n -site replication protocol [7]. An n -way Split could mean encrypting the input data with $n - 1$ keys, or vertically partitioning a database relation into n shares. A Split does not even have to be “cryptographic” – it could represent, say, a vertical partitioning of a database relation into n subsets [1]. While many techniques exist for safeguarding data, the semantics of Copy and Split operators capture a vast majority that are used in practical systems.¹

Secure data storage systems can be modeled as *configurations*, which are compositions of Copy and Split operators. For example, the configuration in Figure 1 describes how a company might safeguard a database containing its trade secrets. The database is represented by the root, r . The root is split into two shares, a and f , say, using encryption. The encrypted database, a , is stored in a public location on the company intranet. Two copies of f (the encryption key) are then made, labeled b and e . One copy, b , is materialized and physically stored, say, on a CD-ROM. The other copy, e , is split again into shares c and d (this time, say, by XOR-ing e with a randomly generated sequence of bits). The terminal vertices a , b , c and d at the bottom of Figure 1 each represent *materialized* data objects. In this example, a is a publicly accessible encrypted version of r , and b , c and d are shares of the encryption key, say, owned by users Bob, Carol and Dave, respectively. The non-terminals e , f , and r represent *transient* data elements that are by-products of the recursive splitting and copying operations. The non-terminals (in particular, the unencrypted database r) are not stored anywhere. Thus, there is no single terminal that can be lost or broken-into that will cause the entire database to be compromised.

Thus, Figure 1 represents a scenario where Bob can decrypt a by himself (perhaps Bob is the company president), but Carol and Dave individually cannot – they must collaborate in order to reconstruct the key f needed to decrypt a (perhaps they are lowly VPs). On the other hand, if one of Bob, Dave or Carol somehow misplaces their share of the encryption key, all is not lost since the key can be reconstructed using the remaining shares. While this example is simplistic, it still shows how a composition of Copy and Split operators

¹The *secret sharing* operator discussed in [6] captures an even broader set of practical techniques (e.g., RAID).

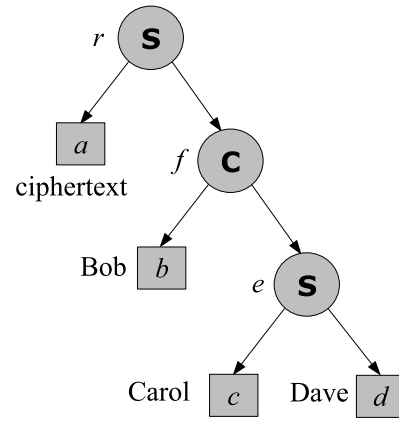


Fig. 1. Example configuration.

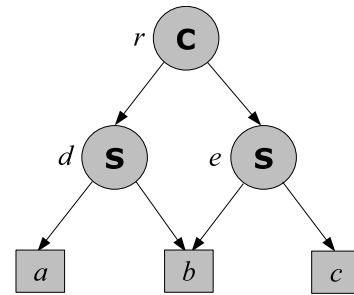


Fig. 2. Configuration with shared vertices.

can simultaneously protect our data from unauthorized access and data loss.

Observe that the root can represent a single database (as in our example), or the entire data storage infrastructure of an organization. In the latter case, an n -way Split at the root might represent a partitioning of data across n sites worldwide, and subsequent layers might describe separate business units, and so on. Since configurations are composable, we can refine our model to any desired granularity. Also note that configurations are not restricted to be trees, but can be rooted directed acyclic graphs (DAGs). For example, consider Figure 2, where d and e are copies of the root data r . Here, the vertex b is shared by both d and e – it might represent a single encryption key that is used to encrypt both d and e . Thus, if Alice, Bob and Carol own data elements a , b and c respectively, then Bob has to collaborate with either Alice or Carol in order to access the decrypted database r . The terminal representing Bob’s key has arrows from both Split operators, since it is needed to reconstruct either copy of r .

In summary, a configuration Θ is a DAG where non-terminal vertices are either Copy or Split operators, and

terminals represent the physical locations across which the copies and shares of the root data are distributed. We denote by \mathcal{T} the set of physical resources (i.e., terminal vertices) in Θ . A *top-down* view of a configuration tells us how copies and shares are generated from the root, and, reversing all arrows so that they point upward, the *bottom-up* view tells us which terminal data elements are needed to reconstruct the original data.

B. Probabilities of Failure

We can use *failure probabilities* to measure the risks faced by a secure data storage system. We define the *probability of break-in*, $P(\Theta)$, as the probability that an attacker breaks into enough terminals in Θ to reconstruct our data. The term “break-in” is used generally to mean any event that leads to an unauthorized party reading our private data e.g., hackers, password leaks, weak passwords, or broken cryptography. Similarly, the *probability of data loss*, $Q(\Theta)$, is defined as the probability that an attacker destroys enough terminals to make reconstruction of our data impossible. “Data loss” refers to events that prevent us from accessing our own data, such as media failures, lost passwords or bit rot.

Consider the configuration illustrated in Figure 2, for example. Let us assume that each of a , b and c is broken-into independently with probability $p = \frac{1}{4}$. An attacker wishing to reconstruct r must do one of three things. He must either break into terminals a and b only, or terminals b and c only, or all three of a , b and c . Thus, the probability of data loss will be the sum of the probabilities of these three mutually exclusively outcomes i.e., $P(\Theta) = 2 \left(\frac{1}{4}\right)^2 \frac{3}{4} + \left(\frac{1}{4}\right)^3 = \frac{7}{64}$. Similarly, an attacker must destroy any of the following sets of terminals to cause r to be lost: $\{b\}$, $\{a, b\}$, $\{b, c\}$, $\{a, c\}$ or $\{a, b, c\}$. Assuming the attacker destroys each terminal independently with probability $q = \frac{1}{4}$, we sum over the probabilities of these five outcomes to find $Q(\Theta) = \frac{1}{4} \left(\frac{3}{4}\right)^2 + 3 \left(\frac{1}{4}\right)^2 \frac{3}{4} + \left(\frac{1}{4}\right)^3 = \frac{19}{64}$.

Although the preceding example assumed independence between the failures of individual terminals, this was purely for illustration. The techniques developed in [5] and [6] can account for arbitrary correlations between the failures of terminals. It is even possible to model correlations between break-in and data loss events e.g., a malicious attacker who destroys all data that he reads (positive correlation), or an intrusion detection system that alerts an administrator upon detecting a break-in (negative correlation).

Thus, the probability that our storage system will succumb to break-ins or data loss is a function of the

failure characteristics of the terminal data elements (i.e., p and q), as well as the manner in which data is distributed across these terminals (i.e., the configuration Θ). The following question arises naturally: Given a set physical resources (i.e., terminals), which configuration will offer us the “best” protection against break-ins and data loss? We address this question next.

C. Optimization

We have described qualitatively that preventing data loss tends to worsen privacy, and preventing break-ins tends to worsen longevity. We can quantify this tradeoff at an atomic level by studying Split and Copy operators. It is easy to show that a Split operator will decrease the probability of break-ins, but increase the probability of data loss, whereas a Copy operator does the exact opposite. Different compositions of Split and Copy operators, therefore, offer us different balances between break-in risk (i.e., $P(\Theta)$) and data loss risk (i.e., $Q(\Theta)$).

It is also easy to show that simply using more physical resources (“throwing more money at the problem”) can simultaneously improve both privacy and longevity i.e., we can achieve arbitrarily low levels of break-in and data loss risk by simply using unbounded numbers of terminals. Therefore, a more meaningful question to ask would be: subject to some maximum level P_0 of break-in risk, and a fixed set \mathcal{T} of physical resources, which is the configuration Θ that minimizes $Q(\Theta)$, the risk of data loss? We express this formally as:

$$\begin{aligned} \min_{\Theta} \quad & Q(\Theta) \\ \text{s.t.} \quad & P(\Theta) \leq P_0 \end{aligned} \tag{1}$$

A solution to (1) would tell us how to optimally use our existing resources in order to minimize our exposure to data loss risk, given our tolerance for break-in risk.² Of course, solving (1) is quite difficult, since the size of the space of configurations can be shown to grow double-exponentially in the number of terminals. The focus of [5], therefore, is to devise efficient solution techniques to (1). A richer version of (1) is formulated and solved in [5], where we can specify arbitrary failure distributions across the terminals, as well as constraints such as which user groups are to be allowed and denied access.

The results of [5] allow us to generate graphs such as the one in Figure 3, which tells us the probability of data

²We could also minimize the break-in risk, $P(\Theta)$, for a fixed level of data loss risk, Q_0 . This “dual problem” is entirely analogous to (1). For simplicity, we focus on (1).

loss we can achieve at various tolerance levels for break-in risk. Figure 3 was generated using four terminals that are each broken-into with probability 20%, and lost with probability 20%, with failures being independent across the terminals. The plot tells us, for instance, that if we can only tolerate a 10% probability of break-in at the root, we can achieve a 13% probability of data-loss. Alternatively, if we can tolerate a 20% probability of break-in, then there is a configuration that achieves a probability of data loss of just 2.7%. This type of graph illustrates how we might sacrifice “a bit” of privacy for a large gain in data longevity.

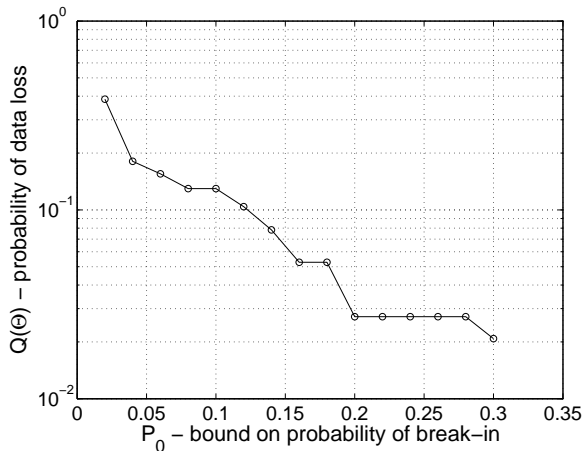


Fig. 3. A privacy-longevity tradeoff curve.

III. INVESTING IN INFORMATION SECURITY

Modeling data storage systems as configurations can also help us make decisions about investing in information security. In particular, we can address two questions:

- How can we better utilize our existing resources?
- How should we allocate our security budget to acquire additional resources?

Suppose we are managing a firm’s data storage infrastructure. We have tolerances P_0 and Q_0 respectively for the break-in and data loss risks, and are given a budget z_0 with which to make necessary investments. Our first step would be to analyze our current system (i.e., write down a configuration Θ that represents our system) and assess the data loss and break-in probabilities.

Now, using \mathcal{T} , P_0 and Q_0 as inputs (recall that \mathcal{T} represents the current set of terminal vertices, or physical resources, in Θ), we can solve (1) to determine if we are using our resources effectively. Is our current configuration offering us protection close to the optimal levels achievable with our current resources? By using

the optimal configuration instead of our current one, it might be possible to achieve (P_0, Q_0) using existing resources. That is, we might achieve our desired levels of risk using resources we already own, without any investment into additional resources.

If we find that (P_0, Q_0) is not attainable using our existing resources, then we need to invest in additional resources. The question then becomes: how should we spend our security budget z_0 ? Simply put, we want to get the most “bang for our buck”. As we will see shortly, our framework also provides guidance here. A recent paper by Gordon and Loeb [4] addressed a similar problem. They ask: what is the optimal level z^* of investment in information security, against a threat that would cause a loss to us of L dollars? The centerpiece of the analysis in [4] is a “security breach probability function”, $S(z, v)$, which tells us the conditional probability of security breach as a result of spending z to defend against the threat, when the current breach probability is v . They leave the exact form of $S(z, v)$ mostly unspecified.

Our framework allows us to actually compute $S(z, v)$ for a given data storage system, as follows. Suppose we had a set \mathcal{I} of possible security technologies we can invest in. For each investment alternative $i \in \mathcal{I}$, we know its cost z_i , and failure probabilities p_i and q_i . For example, $i = 1$ might be an expensive server that is highly resistant to break-ins and data loss (i.e., z_1 is large, p_1 and q_1 are small), whereas $i = 2$ might be a cheaper “commodity” node (i.e., z_2 is smaller, but p_2 and q_2 are larger). We want to find the budget-feasible³ subset $I^* \subseteq \mathcal{I}$ that gives us the largest possible reduction in data loss risk. Then, $S(z_0, v)$ will be the security breach probability that results from investing z_0 into I^* . (Contrary to the assumptions in [4], the function $S(z, v)$ that we compute is not continuous. However, as the authors of [4] acknowledge, any real-world instance of $S(z, v)$ will have discontinuities.)

Formally, let $Q(\Theta)$ be the optimum value of (1), using the current resource set \mathcal{T} . That is, $v = Q(\Theta)$. Let $Q_I(\Theta)$ be the minimum probability of data loss achievable when resource set I is added to our current resource set. We compute $Q_I(\Theta)$ by keeping the break-in risk tolerance P_0 fixed, and solving (1) using $\mathcal{T} \cup I$, rather than just \mathcal{T} . For a fixed z , we then get:

$$S(z, v) = \min_{I \subseteq \mathcal{I} \text{ s.t. } z_I \leq z} Q_I(\Theta) \quad (2)$$

Here, $z_I \equiv \sum_{i \in I} z_i$. Using (2), we can compute $S(z, v)$ for several values of z . Finally, using $S(z, v)$, we can

³Budget-feasibility of I implies $\sum_{i \in I} z_i \leq z_0$.

use the framework in [4] to find the optimum amount z^* to invest in information security.

We have considered here the problem of optimally investing z_0 to reduce data loss risk, while keeping our tolerance for break-in risk fixed. A somewhat different problem is optimally investing to simultaneously reduce break-in risk and data loss risk (rather than keeping break-in risk fixed). There would now be a pair of security breach probability functions, say, $S_{BI}(z, v)$ and $S_{DL}(z, v)$, over which we must optimize. We may need to extend the model in [4] to multiple threat types, and possibly reformulate (1). The fleshing out of such details is a topic for future work.

IV. CONCLUSIONS

We have argued that there is a fundamental tradeoff between ensuring data privacy and data longevity. It is our position that a firm must bear in mind this tradeoff when making design decisions and investment decisions pertaining to security. There are also other considerations, beyond the privacy-longevity tradeoff. For example, the performance, management, and usability of a system are important interrelated factors to consider when choosing a security strategy. The modeling techniques we have developed provide a partial answer to these considerations (e.g., measuring the depth of the configuration, or the total number of operators), but better metrics are clearly needed.

The analysis framework outlined here is a first step toward quantifying the privacy-longevity tradeoff. We feel that the type of structured system modeling presented here can be the basis of economic decision-making in information security contexts.

V. ACKNOWLEDGEMENTS

This work was partially supported by NSERC, and by the NSF through the PORTIA project.

REFERENCES

- [1] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *Proceedings of 2nd Biennial Conference on Innovative Data Systems Research*, 2005.
- [2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.
- [3] D. Boneh, J. Feigenbaum, A. Silberschatz, and R. N. Wright. Portia: Privacy, obligations, and rights in technologies of information assessment. *IEEE Data Engineering Bulletin*, 27, 2004.
- [4] L. A. Gordon and M. P. Loeb. The economics of information security investment. *ACM Transactions on Information and System Security*, 20(5):438–457, Nov. 2002.

- [5] B. Mungamuru, H. Garcia-Molina, and S. Mitra. How to safeguard your sensitive data. In *Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems*, 2006.
- [6] B. Mungamuru, H. Garcia-Molina, and C. Olston. Configurations: Understanding alternatives for safeguarding data. *Stanford InfoLab Technical Report*, 2005. <http://dbpubs.stanford.edu/pub/2005-41>.
- [7] V. Reich and D. S. H. Rosenthal. LOCKSS: Lots of copies keeps stuff safe. In *Proceedings of Preservation 2000*, 2000.
- [8] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.
- [9] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.
- [10] J. Wylie, M. Bigrigg, J. Strunk, G. Ganger, H. Kiliccote, and P. Khosla. Survivable information storage systems. *IEEE Computer*, 33(8):61–68, Aug. 2000.