How to write a scientific article

Hyosang Kang

Jeonbuk Science High School

21 July, 2023

What is a scientific article?

A scientific article is a written document that often describes the process, results, and conclusions of a scientific research. It is a way to publicize your research output and communicate within the scientific community.

What is a scientific writing?

A scientific writing is a skill for writing a scientific article. A scientific article is usually written in a format of a journal paper which requires a specific structure and style.

Why do we need scientific writing?

In this document, we will discuss about how to write a scientific article properly. We will also discuss how scientific writing should be done with hand-on examples. At the end of our discussion, you will understand a good scientific writing skill helps not only producing a good scientific article, but also accomplishing a successful research.

Who are the audiences?

This document is designed for a special lecture to students who begin to write their first scientific article. The author assumes the range of audiences are from senior high school students to undergraduate students, specifically in South Korea.

Why am I write this document in English?

By the way, you may wonder why this document is written in English. First, most of scientific articles are written in English, so you must get used to English writing whether you like it or not. Second, a tranlating Korean contents to English is not a good way of writing a scientific article, because there is no one-to-one mapping between Korean and English. You will end up with either an awkward English translation or rewriting the whole contents in English from scratch.

Let's group up!

Before we start, let's group up by three (or four) people. Each group will have a writer, a reporter, and a interpreter. There may be two interpreter depending on the size of your group.

The roles of the group members

The group members should actively participate in the discussion of the topics in the following sections. The writer will write down the conclusion of the group in this piece of paper. The reporter will report the conclusion of the group to the class when called by the lecturer. The interpreter will rephrase, elaborate, criticize, or summarize the arguments in the group.
Once you have decided the roles of the group members, please attach a sticker to the chest of the reporter so that the lecturer can see who is the reporter in the group.

Sections of a scientific article

Every scientific article is composed of several sections. Sections are organized in a way that the reader can easily follow the context of the article. Using sections, the writer can also easily organize the contents of the article. Let's discuss about what kinds of sections are there.

In the appendix, you can find two sample scientific articles listed below:

- **Brunton S.L. et al.** (2016) Discovering governing equations from data by sparse identification of nonlinear dynamical systems. PNAS, 113(15), 3932-3937.
- **Rivest R.L. et al.** (1978) A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

You should not attempt to read the whole document. Instead, quickly skim through the document and try to divide the contents into distinct sections. While doing so, try to name the sections. (e.g. Abstract, Introduction, Method, Results, Conclusion, etc.)

Here is the list of questions you should discuss in your group:

1. What are names of sections (not the titles!) you have found in two sample articles?
2. What do you think the role of each section is?
3. What are the common sections in two sample articles?
4. What are the differences in sections between two sample articles?

The order of writing the sections

Now suppose you are about to write a scientific article from scratch. You opened your favorite text editor and are ready to write the first sentence. What should you write first?

Here is the list of questions you should discuss in your group:

1. From the list of sections you have found previously, what section should you write first, and why?
2. In what order would you write the rest of the sections, and why?
3. In which section, would you spend most of your time in writing and why?

What, why, and how

When we make a scientific argument, we have a concrete form of a question in our mind, and this question is either what, why, or how question.

A what question is something like: What is *thing* that we are going to talking about? What is the problem about that *thing*? What have people done about that *thing*?

A why question is something like: Why is studying that *thing* important? Why is solving our problem important? Why is that our problem difficult to solve?

A how question is something like: How are we going to solve our problem? How are we going to prove that our solution is correct? How are we going to evaluate our solution?

Writing is a dialogue between the writer and the reader. Thus writer should keep the readers focused by laying out a series of answers to the questions that the reader might have in mind while reading the document. In the scientific writing, authors should set the level of the audience clearly so that they can project the questions onto their audience properly. Once the questions are properly projected, the rest of the work is to write a truthful, factual, objective, and comprehensive answers to the questions.

Let's do an exercise to classify the sentences in the sample articles into answers to what, why, and how questions.

1. Divide the sentences in the sample articles into three groups: what, why, and how.
   - From the first page of **Brunton S.L. et al.** (2016)

> Advances in machine learning (Ref. #1) and data science (Ref. #22) have promised a renaissance in the analysis and understanding of complex data, extracting patterns in vast multimodal data that are beyond the ability of humans to grasp. However, despite the rapid development of tools to understand static data based on statistical relationships, there has been slow progress in distilling physical models of dynamic processes from big data. This has limited the ability of data science models to extrapolate the dynamics beyond the attractor where they were sampled and constructed.

- From the first page of **Rivest R.L. et al.** (1978)

> An encryption (or decryption) procedure typically consists of a general method and an encryption key. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the ciphertext C. Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

2. Write down a concrete question for each sentence that the sentence is the answer for.
   - Advances in machine learning (Ref. #1) and data science (Ref. #22) have promised a renaissance in the analysis and understanding of complex data, extracting patterns in vast multimodal data that are beyond the ability of humans to grasp.
   - However, despite the rapid development of tools to understand static data based on statistical relationships, there has been slow progress in distilling physical models of dynamic processes from big data.
   - This has limited the ability of data science models to extrapolate the dynamics beyond the attractor where they were sampled and constructed.
3. (Homework) Do the same with the sentences in the second box above.

Few tips for improving your scientific writing skill

Here are some tips for improving your scientific writing skill.

1. Give a plenty of time to write, and also for the revision. Start writing at the beginning of your research, not at the end.
2. Make a claim only with a concrete logical argument, or with a credible reference (for example, wiki is not a credible source.)
3. Avoid using any subjective adjectives (e.g. important, significant, etc.) or adverbs (e.g. very, extremely, highly, etc.)
4. Remove redundencies in your sentences, paragraphs, and sections. (Unfortunately, creating redundancy is what text-generating AIs are extremely good at! So do not rely on them too much.)
5. Use as many writing assistant tools as possible (e.g. spell checker, grammar checker, etc.) Learn how to use LaTeX(pronounced as "lay-tech") or any other word processor that supports mathematical equations. Hangul® is unaccepted as an official word processor in world standard.

Wrapping up

Before we finish, let's discuss about the following things among your group members:

1. What have you learned from today's lecture?
2. What was the thing that highlighted you the most, and why do you think so?
3. What would you do differently in your scientific writing than you thought to do (or already have been doing) before this lecture?

Discovering governing equations from data by sparse identification of nonlinear dynamical systems

Steven L. Brunton[a,1], Joshua L. Proctor[b], and J. Nathan Kutz[c]

[a]Department of Mechanical Engineering, University of Washington, Seattle, WA 98195; [b]Institute for Disease Modeling, Bellevue, WA 98005; and [c]Department of Applied Mathematics, University of Washington, Seattle, WA 98195

Edited by William Bialek, Princeton University, Princeton, NJ, and approved March 1, 2016 (received for review August 31, 2015)

Extracting governing equations from data is a central challenge in many diverse areas of science and engineering. Data are abundant whereas models often remain elusive, as in climate science, neuroscience, ecology, finance, and epidemiology, to name only a few examples. In this work, we combine sparsity-promoting techniques and machine learning with nonlinear dynamical systems to discover governing equations from noisy measurement data. The only assumption about the structure of the model is that there are only a few important terms that govern the dynamics, so that the equations are sparse in the space of possible functions; this assumption holds for many physical systems in an appropriate basis. In particular, we use sparse regression to determine the fewest terms in the dynamic governing equations required to accurately represent the data. This results in parsimonious models that balance accuracy with model complexity to avoid overfitting. We demonstrate the algorithm on a wide range of problems, from simple canonical systems, including linear and nonlinear oscillators and the chaotic Lorenz system, to the fluid vortex shedding behind an obstacle. The fluid example illustrates the ability of this method to discover the underlying dynamics of a system that took experts in the community nearly 30 years to resolve. We also show that this method generalizes to parameterized systems and systems that are time-varying or have external forcing.

dynamical systems | machine learning | sparse regression | system identification | optimization

Advances in machine learning (1) and data science (2) have promised a renaissance in the analysis and understanding of complex data, extracting patterns in vast multimodal data that are

dynamical systems from data. However, symbolic regression is expensive, does not scale well to large systems of interest, and may be prone to overfitting unless care is taken to explicitly balance model complexity with predictive power. In ref. 4, the Pareto front is used to find parsimonious models. There are other techniques that address various aspects of the dynamical system discovery problem. These include methods to discover governing equations from time-series data (6), equation-free modeling (7), empirical dynamic modeling (8, 9), modeling emergent behavior (10), and automated inference of dynamics (11–13); ref. 12 provides an excellent review.

Sparse Identification of Nonlinear Dynamics (SINDy)

In this work, we reenvision the dynamical system discovery problem from the perspective of sparse regression (14–16) and compressed sensing (17–22). In particular, we leverage the fact that most physical systems have only a few relevant terms that define the dynamics, making the governing equations sparse in a high-dimensional nonlinear function space. The combination of sparsity methods in dynamical systems is quite recent (23–30). Here, we consider dynamical systems (31) of the form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)). \quad [1]$$

The vector $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state of a system at time t, and the function $\mathbf{f}(\mathbf{x}(t))$ represents the dynamic constraints that define the equations of motion of the system, such as Newton's second law. Later, the dynamics will be generalized to include parameterization, time dependence, and forcing.

the ability of this method to discover the underlying dynamics of a system that took experts in the community nearly 30 years to resolve. We also show that this method generalizes to parameterized systems and systems that are time-varying or have external forcing.

dynamical systems | machine learning | sparse regression | system identification | optimization

Advances in machine learning (1) and data science (2) have promised a renaissance in the analysis and understanding of complex data, extracting patterns in vast multimodal data that are beyond the ability of humans to grasp. However, despite the rapid development of tools to understand static data based on statistical relationships, there has been slow progress in distilling physical models of dynamic processes from big data. This has limited the ability of data science models to extrapolate the dynamics beyond the attractor where they were sampled and constructed.

An analogy may be drawn with the discoveries of Kepler and Newton. Kepler, equipped with the most extensive and accurate planetary data of the era, developed a data-driven model for planetary motion, resulting in his famous elliptic orbits. However, this was an attractor-based view of the world, and it did not explain the fundamental dynamic relationships that give rise to planetary orbits, or provide a model for how these bodies react when perturbed. Newton, in contrast, discovered a dynamic relationship between momentum and energy that described the underlying processes responsible for these elliptic orbits. This dynamic model may be generalized to predict behavior in regimes where no data were collected. Newton's model has proven remarkably robust for engineering design, making it possible to land a spacecraft on the moon, which would not have been possible using Kepler's model alone.

A seminal breakthrough by Bongard and Lipson (3) and Schmidt and Lipson (4) has resulted in a new approach to determine the underlying structure of a nonlinear dynamical system from data. This method uses symbolic regression [i.e., genetic programming (5)] to find nonlinear differential equations, and it balances complexity of the model, measured in the number of terms, with model accuracy. The resulting model identification realizes a long-sought goal of the physics and engineering communities to discover high-dimensional nonlinear function space. The combination of sparsity methods in dynamical systems is quite recent (23–30). Here, we consider dynamical systems (31) of the form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)). \quad [1]$$

The vector $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state of a system at time t, and the function $\mathbf{f}(\mathbf{x}(t))$ represents the dynamic constraints that define the equations of motion of the system, such as Newton's second law. Later, the dynamics will be generalized to include parameterization, time dependence, and forcing.

Significance

Understanding dynamic constraints and balances in nature has facilitated rapid development of knowledge and enabled technology, including aircraft, combustion engines, satellites, and electrical power. This work develops a novel framework to discover governing equations underlying a dynamical system simply from data measurements, leveraging advances in sparsity techniques and machine learning. The resulting models are parsimonious, balancing model complexity with descriptive ability while avoiding overfitting. There are many critical data-driven problems, such as understanding cognition from neural recordings, inferring climate patterns, determining stability of financial markets, predicting and suppressing the spread of disease, and controlling turbulence for greener transportation and energy. With abundant data and elusive laws, data-driven discovery of dynamics will continue to play an important role in these efforts.

Author contributions: S.L.B., J.L.P., and J.N.K. designed research; S.L.B. performed research; S.L.B., J.L.P., and J.N.K. analyzed data; and S.L.B. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

[1]To whom correspondence should be addressed. Email: sbrunton@uw.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1517384113/-/DCSupplemental.

The key observation is that for many systems of interest, the function $\mathbf{f}$ consists of only a few terms, making it sparse in the space of possible functions. Recent advances in compressed sensing and sparse regression make this viewpoint of sparsity favorable, because it is now possible to determine which right-hand-side terms are nonzero without performing a combinatorially intractable brute-force search. This guarantees that the sparse solution is found with high probability using convex methods that scale to large problems favorably with Moore's law. The resulting sparse model identification inherently balances model complexity (i.e., sparsity of the right-hand-side dynamics) with accuracy, avoiding overfitting the model to the data. Wang et al. (23) have used compressed sensing to identify nonlinear dynamics and predict catastrophes; here, we advocate using sparse regression to mitigate noise.

To determine the function $\mathbf{f}$ from data, we collect a time history of the state $\mathbf{x}(t)$ and either measure the derivative $\dot{\mathbf{x}}(t)$ or approximate it numerically from $\mathbf{x}(t)$. The data are sampled at several times $t_1, t_2, \cdots, t_m$ and arranged into two matrices:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \overrightarrow{\text{state}} \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \downarrow \text{time}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix}.$$

Next, we construct a library $\Theta(\mathbf{X})$ consisting of candidate nonlinear functions of the columns of $\mathbf{X}$. For example, $\Theta(\mathbf{X})$ may consist of constant, polynomial, and trigonometric terms:

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & \mathbf{X} & \mathbf{X}^{P_2} & \mathbf{X}^{P_3} & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix}. \quad [2]$$

Here, higher polynomials are denoted as $\mathbf{X}^{P_2}, \mathbf{X}^{P_3}$, etc., where

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \Xi^T \left(\Theta\left(\mathbf{x}^T\right)\right)^T. \quad [5]$$

Each column of Eq. **3** requires a distinct optimization to find the sparse vector of coefficients $\boldsymbol{\xi}_k$ for the kth row equation. We may also normalize the columns of $\Theta(\mathbf{X})$, especially when entries of $\mathbf{X}$ are small, as discussed in the *SI Appendix*.

For examples in this paper, the matrix $\Theta(\mathbf{X})$ has size $m \times p$, where p is the number of candidate functions, and $m \gg p$ because there are more data samples than functions; this is possible in a restricted basis, such as the polynomial basis in Eq. **2**. In practice, it may be helpful to test many different function bases and use the sparsity and accuracy of the resulting model as a diagnostic tool to determine the correct basis to represent the dynamics in. In *SI Appendix*, Appendix B, two examples are explored where the sparse identification algorithm fails because the dynamics are not sparse in the chosen basis.

Realistically, often only $\mathbf{X}$ is available, and $\dot{\mathbf{X}}$ must be approximated numerically, as in all of the continuous-time examples below. Thus, $\mathbf{X}$ and $\dot{\mathbf{X}}$ are contaminated with noise so Eq. **3** does not hold exactly. Instead,

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi + \eta\mathbf{Z}, \quad [6]$$

where $\mathbf{Z}$ is modeled as a matrix of independent identically distributed Gaussian entries with zero mean, and noise magnitude η. Thus, we seek a sparse solution to an overdetermined system with noise. The least absolute shrinkage and selection operator (LASSO) (14, 15) is an ℓ_1-regularized regression that promotes sparsity and works well with this type of data. However, it may be computationally expensive for very large data sets. An alternative based on sequential thresholded least-squares is presented in Code 1 in the *SI Appendix*.

Depending on the noise, it may be necessary to filter $\mathbf{X}$ and $\dot{\mathbf{X}}$ before solving for Ξ. In many of the examples below, only the data $\mathbf{X}$ are available, and $\dot{\mathbf{X}}$ are obtained by differentiation. To counteract differentiation error, we use the total variation regularization (32) to denoise the derivative (33). This works quite well when only state data $\mathbf{X}$ are available, as illustrated on the Lorenz system (*SI Appendix*, Fig. S7). Alternatively, the data $\mathbf{X}$ and $\dot{\mathbf{X}}$ may be filtered, for example using the optimal hard threshold for singular values described in ref. 34. Insensitivity to noise is a critical feature of an

$$\begin{bmatrix} \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} \quad \begin{bmatrix} \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix}$$

Next, we construct a library $\Theta(\mathbf{X})$ consisting of candidate nonlinear functions of the columns of $\mathbf{X}$. For example, $\Theta(\mathbf{X})$ may consist of constant, polynomial, and trigonometric terms:

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & \mathbf{X} & \mathbf{X}^{P_2} & \mathbf{X}^{P_3} & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix}. \quad [2]$$

Here, higher polynomials are denoted as $\mathbf{X}^{P_2}, \mathbf{X}^{P_3}$, etc., where $\mathbf{X}^{P_2}$ denotes the quadratic nonlinearities in the state $\mathbf{x}$:

$$\mathbf{X}^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & \cdots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & \cdots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \cdots & x_2^2(t_m) & \cdots & x_n^2(t_m) \end{bmatrix}.$$

Each column of $\Theta(\mathbf{X})$ represents a candidate function for the right-hand side of Eq. **1**. There is tremendous freedom in choosing the entries in this matrix of nonlinearities. Because we believe that only a few of these nonlinearities are active in each row of $\mathbf{f}$, we may set up a sparse regression problem to determine the sparse vectors of coefficients $\Xi = [\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n]$ that determine which nonlinearities are active:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi. \quad [3]$$

This is illustrated in Fig. 1. Each column $\boldsymbol{\xi}_k$ of Ξ is a sparse vector of coefficients determining which terms are active in the right-hand side for one of the row equations $\dot{\mathbf{x}}_k = \mathbf{f}_k(\mathbf{x})$ in Eq. **1**. Once Ξ has been determined, a model of each row of the governing equations may be constructed as follows:

$$\dot{\mathbf{x}}_k = \mathbf{f}_k(\mathbf{x}) = \Theta(\mathbf{x}^T)\boldsymbol{\xi}_k. \quad [4]$$

Note that $\Theta(\mathbf{x}^T)$ is a vector of symbolic functions of elements of $\mathbf{x}$, as opposed to $\Theta(\mathbf{X})$, which is a data matrix. Thus,

computationally expensive for very large data sets. An alternative based on sequential thresholded least-squares is presented in Code 1 in the *SI Appendix*.

Depending on the noise, it may be necessary to filter $\mathbf{X}$ and $\dot{\mathbf{X}}$ before solving for Ξ. In many of the examples below, only the data $\mathbf{X}$ are available, and $\dot{\mathbf{X}}$ are obtained by differentiation. To counteract differentiation error, we use the total variation regularization (32) to denoise the derivative (33). This works quite well when only state data $\mathbf{X}$ are available, as illustrated on the Lorenz system (*SI Appendix*, Fig. S7). Alternatively, the data $\mathbf{X}$ and $\dot{\mathbf{X}}$ may be filtered, for example using the optimal hard threshold for singular values described in ref. 34. Insensitivity to noise is a critical feature of an algorithm that identifies dynamics from data (11–13).

Often, the physical system of interest may be naturally represented by a partial differential equation (PDE) in a few spatial variables. If data are collected from a numerical discretization or from experimental measurements on a spatial grid, then the state dimension n may be prohibitively large. For example, in fluid dynamics, even simple 2D and 3D flows may require tens of thousands up to billions of variables to represent the discretized system. The proposed method is ill-suited for a large state dimension n, because of the factorial growth of Θ in n and because each of the n row equations in Eq. **4** requires a separate optimization. Fortunately, many high-dimensional systems of interest evolve on a low-dimensional manifold or attractor that is well-approximated using a low-rank basis Ψ (35, 36). For example, if data $\mathbf{X}$ are collected for a high-dimensional system as in Eq. **2**, it is possible to obtain a low-rank approximation using dimensionality reduction techniques, such as the proper orthogonal decomposition (POD) (35, 37).

The proposed sparse identification of nonlinear dynamics (SINDy) method depends on the choice of measurement variables, data quality, and the sparsifying function basis. There is no single method that will solve all problems in nonlinear system identification, but this method highlights the importance of these underlying choices and can help guide the analysis. The challenges of choosing measurement variables and a sparsifying function basis are explored in *SI Appendix*, section 4.5 and Appendixes A and B.

Simply put, we need the right coordinates and function basis to yield sparse dynamics; the feasibility and flexibility of these requirements is discussed in *Discussion* and *SI Appendix* section 4.5

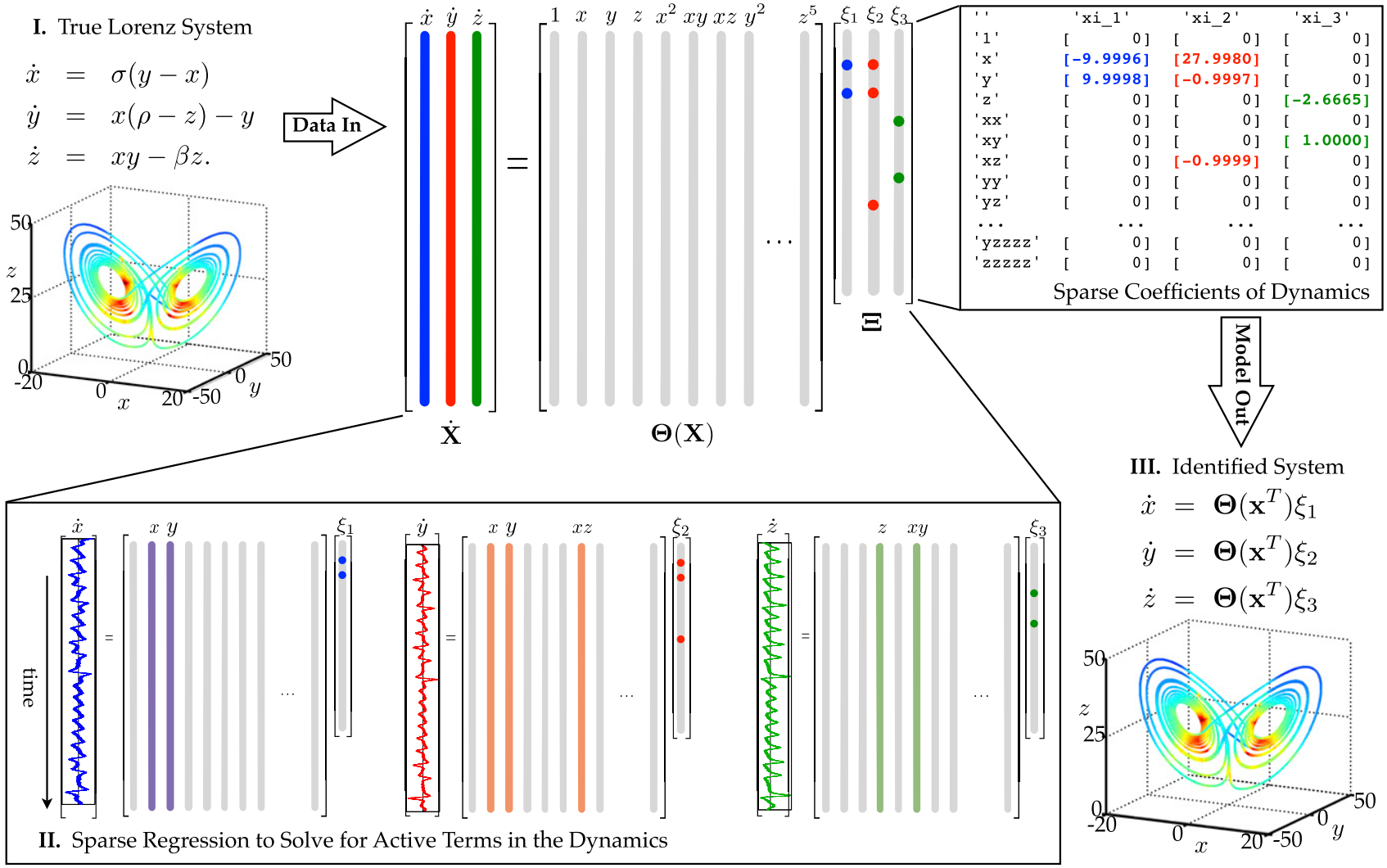

Fig. 1. Schematic of the SINDy algorithm, demonstrated on the Lorenz equations. Data are collected from the system, including a time history of the states $\mathbf{X}$ and derivatives $\dot{\mathbf{X}}$; the assumption of having $\dot{\mathbf{X}}$ is relaxed later. Next, a library of nonlinear functions of the states, $\Theta(\mathbf{X})$, is constructed. This nonlinear feature library is used to find the fewest terms needed to satisfy $\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi$. The few entries in the vectors of Ξ, solved for by sparse regression, denote the relevant terms in the right-hand side of the dynamics. Parameter values are $\sigma = 10, \beta = 8/3, \rho = 28$, $(x_0, y_0, z_0)^T = (-8, 7, 27)^T$. The trajectory on the Lorenz attractor is colored by the adaptive time step required, with red indicating a smaller time step.

and Appendixes A and B. However, it may be difficult to know the correct variables a priori. Fortunately, time-delay coordi-

recovering the parameterized logistic map and the Hopf normal form from noisy measurements. In each example, we explore the

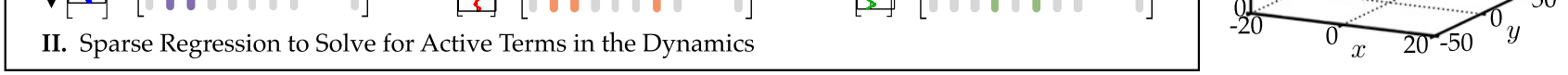

Fig. 1. Schematic of the SINDy algorithm, demonstrated on the Lorenz equations. Data are collected from the system, including a time history of the states $\mathbf{X}$ and derivatives $\dot{\mathbf{X}}$; the assumption of having $\dot{\mathbf{X}}$ is relaxed later. Next, a library of nonlinear functions of the states, $\Theta(\mathbf{X})$, is constructed. This nonlinear feature library is used to find the fewest terms needed to satisfy $\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$. The few entries in the vectors of $\boldsymbol{\Xi}$, solved for by sparse regression, denote the relevant terms in the right-hand side of the dynamics. Parameter values are $\sigma = 10, \beta = 8/3, \rho = 28$, $(x_0, y_0, z_0)^T = (-8, 7, 27)^T$. The trajectory on the Lorenz attractor is colored by the adaptive time step required, with red indicating a smaller time step.

and Appendixes A and B. However, it may be difficult to know the correct variables a priori. Fortunately, time-delay coordinates may provide useful variables from a time series (9, 12, 38). The ability to reconstruct sparse attractor dynamics using time-delay coordinates is demonstrated in *SI Appendix*, section 4.5 using a single variable of the Lorenz system.

The choice of coordinates and the sparsifying basis are intimately related, and the best choice is not always clear. However, basic knowledge of the physics (e.g., Navier–Stokes equations have quadratic nonlinearities, and the Schrödinger equation has $|\mathbf{x}|^2\mathbf{x}$ terms) may provide a reasonable choice of nonlinear functions and measurement coordinates. In fact, the sparsity and accuracy of the proposed sparse identified model may provide valuable diagnostic information about the correct measurement coordinates and basis in which to represent the dynamics. Choosing the right coordinates to simplify dynamics has always been important, as exemplified by Lagrangian and Hamiltonian mechanics (39). There is still a need for experts to find and exploit symmetry in the system, and the proposed methods should be complemented by advanced algorithms in machine learning to extract useful features.

Results

We demonstrate the algorithm on canonical systems*, ranging from linear and nonlinear oscillators (*SI Appendix*, section 4.1), to noisy measurements of the chaotic Lorenz system, to the unsteady fluid wake behind a cylinder, extending this method to nonlinear PDEs and high-dimensional data. Finally, we show that bifurcation parameters may be included in the models, recovering the parameterized logistic map and the Hopf normal form from noisy measurements. In each example, we explore the ability to identify the dynamics from state measurements alone, without access to derivatives.

It is important to reiterate that the sparse identification method relies on a fortunate choice of coordinates and function basis that facilitate sparse representation of the dynamics. In *SI Appendix*, Appendix B, we explore the limitations of the method for examples where these assumptions break down: the Lorenz system transformed into nonlinear coordinates and the glycolytic oscillator model (11–13).

Chaotic Lorenz System. As a first example, consider a canonical model for chaotic dynamics, the Lorenz system (40):

$$\dot{x} = \sigma(y - x), \qquad [7a]$$

$$\dot{y} = x(\rho - z) - y, \qquad [7b]$$

$$\dot{z} = xy - \beta z. \qquad [7c]$$

Although these equations give rise to rich and chaotic dynamics that evolve on an attractor, there are only a few terms in the right-hand side of the equations. Fig. 1 shows a schematic of how data are collected for this example, and how sparse dynamics are identified in a space of possible right-hand-side functions using convex ℓ_1 minimization.

For this example, data are collected for the Lorenz system, and stacked into two large data matrices $\mathbf{X}$ and $\dot{\mathbf{X}}$, where each row of $\mathbf{X}$ is a snapshot of the state $\mathbf{x}$ in time, and each row of $\dot{\mathbf{X}}$ is a snapshot

*Code is available at faculty.washington.edu/sbrunton/sparsedynamics.zip.

of the time derivative of the state $\dot{\mathbf{x}}$ in time. Here, the right-hand-side dynamics are identified in the space of polynomials $\Theta(\mathbf{X})$ in (x,y,z) up to fifth order, although other functions such as sin, cos, exp, or higher-order polynomials may be included:

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & | & | & \cdots & | \\ x(t) & \mathbf{y}(t) & \mathbf{z}(t) & \mathbf{x}(t)^2 & \mathbf{x}(t)\mathbf{y}(t) & \cdots & \mathbf{z}(t)^5 \\ | & | & | & | & | & \cdots & | \end{bmatrix}.$$

Each column of $\Theta(\mathbf{X})$ represents a candidate function for the right-hand side of Eq. **1**. Because only a few of these terms are active in each row of $\mathbf{f}$, we solve the sparse regression problem in Eq. **3** to determine the sparse vectors of coefficients $\Xi = [\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2 \cdots \boldsymbol{\xi}_n]$ that determine which terms are active in the dynamics. This is illustrated schematically in Fig. 1, where sparse vectors $\boldsymbol{\xi}_k$ are found to represent the derivative $\dot{x}_k$ as a linear combination of the fewest terms in $\Theta(\mathbf{X})$.

In the Lorenz example, the ability to capture dynamics on the attractor is more important than the ability to predict an individual trajectory, because chaos will quickly cause any small variations in initial conditions or model coefficients to diverge exponentially. As shown in Fig. 1, the sparse model accurately reproduces the attractor dynamics from chaotic trajectory measurements. The algorithm not only identifies the correct terms in the dynamics, but it accurately determines the coefficients to within .03% of the true values. We also explore the identification of the dynamics when only noisy state measurements are available (*SI Appendix*, Fig. S7). The correct dynamics are identified, and the attractor is preserved for surprisingly large noise values. In *SI Appendix*, section 4.5, we reconstruct the attractor dynamics in the Lorenz system using time-delay coordinates from a single measurement $x(t)$.

PDE for Vortex Shedding Behind an Obstacle. The Lorenz system is a low-dimensional model of more realistic high-dimensional PDE models for fluid convection in the atmosphere. Many systems of interest are governed by PDEs (24), such as weather and climate, epidemiology, and the power grid, to name a few. Each of these examples is characterized by big data, consisting of large spatially resolved measurements consisting of millions or billions of states and spanning orders of magnitude of scale in both space and time. However, many high-dimensional, real-world systems evolve on a low-dimensional attractor, making the effective dimension much smaller (35)

(41–43). Data are collected for the fluid flow past a cylinder at Reynolds number 100 using direct numerical simulations of the 2D Navier–Stokes equations (44, 45). The nonlinear dynamic relationship between the dominant coherent structures is identified from these flow-field measurements with no knowledge of the governing equations.

The flow past a cylinder is a particularly interesting example because of its rich history in fluid mechanics and dynamical systems. It has long been theorized that turbulence is the result of a series of Hopf bifurcations that occur as the flow velocity increases (46), giving rise to more rich and intricate structures in the fluid. After 15 years, the first Hopf bifurcation was discovered in a fluid system, in the transition from a steady laminar wake to laminar periodic vortex shedding at Reynolds number 47 (47, 48). This discovery led to a long-standing debate about how a Hopf bifurcation, with cubic nonlinearity, can be exhibited in a Navier–Stokes fluid with quadratic nonlinearities. After 15 more years, this was resolved using a separation of timescales and a mean-field model (49), shown in Eq. **8**. It was shown that coupling between oscillatory modes and the base flow gives rise to a slow manifold (Fig. 2, *Left*), which results in algebraic terms that approximate cubic nonlinearities on slow timescales.

This example provides a compelling test case for the proposed algorithm, because the underlying form of the dynamics took nearly three decades for experts in the community to uncover. Because the state dimension is large, consisting of the fluid state at hundreds of thousands of grid points, it is first necessary to reduce the dimension of the system. The POD (35, 37), provides a low-rank basis resulting in a hierarchy of orthonormal modes that, when truncated, capture the most energy of the original system for the given rank truncation. The first two most energetic POD modes capture a significant portion of the energy, and steady-state vortex shedding is a limit cycle in these coordinates. An additional mode, called the shift mode, is included to capture the transient dynamics connecting the unstable steady state ("C" in Fig. 2) with the mean of the limit cycle (49) ("B" in Fig. 2). These modes define the x,y,z coordinates in Fig. 2.

In the coordinate system described above, the mean-field model for the cylinder dynamics is given by (49)

$$\dot{x} = \mu x - \omega y + Axz, \qquad \textbf{[8a]}$$

$$\dot{y} = \omega x + \mu y + Ayz, \qquad \textbf{[8b]}$$

time-delay coordinates from a single measurement $x(t)$.

PDE for Vortex Shedding Behind an Obstacle. The Lorenz system is a low-dimensional model of more realistic high-dimensional PDE models for fluid convection in the atmosphere. Many systems of interest are governed by PDEs (24), such as weather and climate, epidemiology, and the power grid, to name a few. Each of these examples is characterized by big data, consisting of large spatially resolved measurements consisting of millions or billions of states and spanning orders of magnitude of scale in both space and time. However, many high-dimensional, real-world systems evolve on a low-dimensional attractor, making the effective dimension much smaller (35).

Here we generalize the SINDy method to an example in fluid dynamics that typifies many of the challenges outlined above. In the context of data from a PDE, our algorithm shares some connections to the dynamic mode decomposition, which is a linear dynamic regression

steady-state vortex shedding is a limit cycle in these coordinates. An additional mode, called the shift mode, is included to capture the transient dynamics connecting the unstable steady state ("C" in Fig. 2) with the mean of the limit cycle (49) ("B" in Fig. 2). These modes define the x,y,z coordinates in Fig. 2.

In the coordinate system described above, the mean-field model for the cylinder dynamics is given by (49)

$$\dot{x} = \mu x - \omega y + Axz, \qquad \textbf{[8a]}$$

$$\dot{y} = \omega x + \mu y + Ayz, \qquad \textbf{[8b]}$$

$$\dot{z} = -\lambda\left(z - x^2 - y^2\right). \qquad \textbf{[8c]}$$

If λ is large, so that the z dynamics are fast, then the mean flow rapidly corrects to be on the (slow) manifold $z = x^2 + y^2$ given by

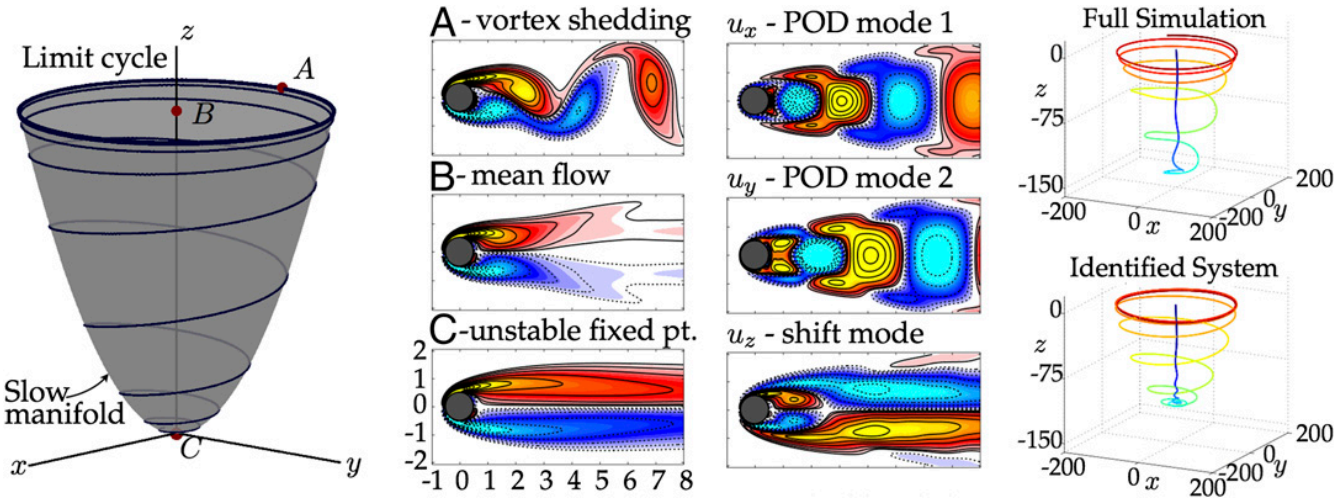

Fig. 2. Example of high-dimensional dynamical system from fluid dynamics. The vortex shedding past a cylinder is a prototypical example that is used for flow control, with relevance to many applications, including drag reduction behind vehicles. The vortex shedding is the result of a Hopf bifurcation. However, because the Navier–Stokes equations have quadratic nonlinearity, it is necessary to use a mean-field model with a separation of timescales, where a fast mean-field deformation is slave to the slow vortex shedding dynamics. The parabolic slow manifold is shown (*Left*), with the unstable fixed point (*C*), mean flow (*B*), and vortex shedding (*A*). A POD basis and shift mode are used to reduce the dimension of the problem (*Middle Right*). The identified dynamics closely match the true trajectory in POD coordinates, and most importantly, they capture the quadratic nonlinearity and timescales associated with the mean-field model.

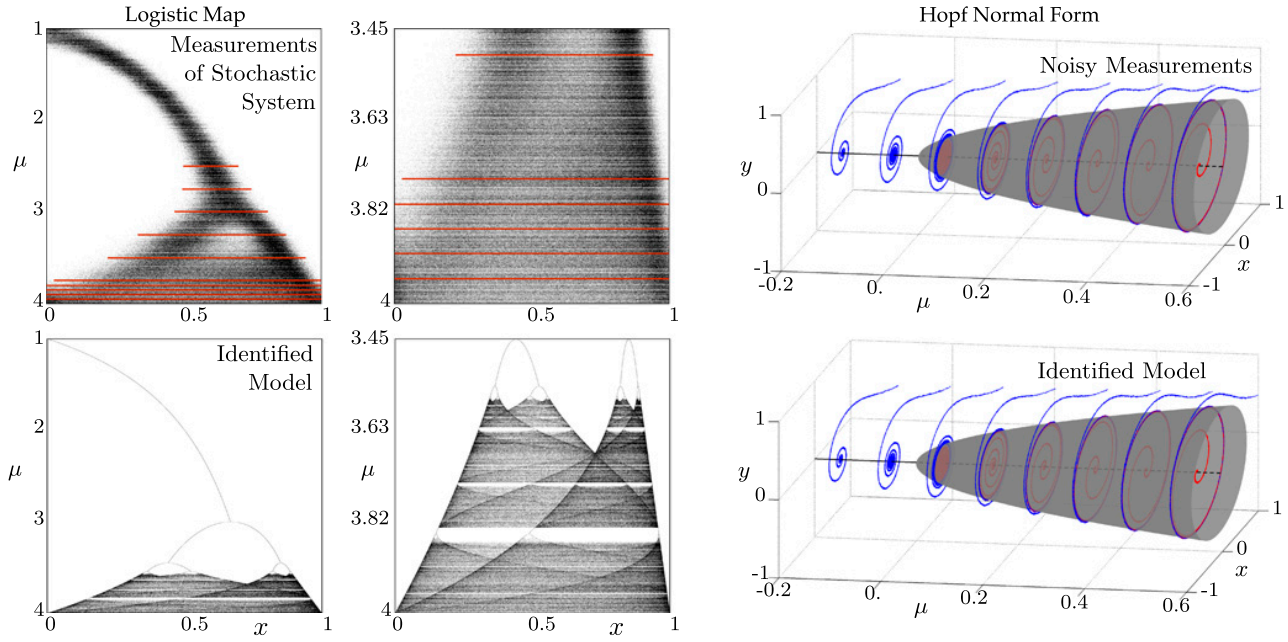

Fig. 3. SINDy algorithm is able to identify normal forms and capture bifurcations, as demonstrated on the logistic map (*Left*) and the Hopf normal form (*Right*). Noisy data from both systems are used to train models. For the logistic map, a handful of parameter values μ (red lines), are used for the training data, and the correct normal form and bifurcation sequence is captured (below). Noisy data for the Hopf normal form are collected at a few values of μ, and the total variation derivative (33) is used to compute time derivatives. The accurate Hopf normal form is reproduced (below). The nonlinear terms identified by the algorithm are in *SI Appendix*, section 4.4 and Appendix C.

the amplitude of vortex shedding. When substituting this algebraic relationship into Eqs. **8a** and **8b**, we recover the Hopf normal form on the slow manifold.

With a time history of these three coordinates, the proposed algorithm correctly identifies quadratic nonlinearities and reproduces a parabolic slow manifold. Note that derivative measurements are not available, but are computed from the state variables. Interestingly, when the training data do not include trajectories that originate off of the slow manifold, the algorithm incorrectly identifies cubic nonlinearities, and fails to identify the slow manifold.

Normal Forms, Bifurcations, and Parameterized Systems. In practice, many real-world systems depend on parameters, and dramatic changes, or bifurcations, may occur when the parameter is varied. The SINDy algorithm is readily extended to encompass these important parameterized systems, allowing for the discovery of normal forms (31, 50) associated with a bifurcation parameter μ. First, we append μ to the dynamics:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}; \mu), \qquad [9a]$$

In the Hopf normal-form example, noisy state measurements from eight parameter values are sampled, with data collected on the blue and red trajectories in Fig. 3 (*Top Right*). Noise is added to the position measurements to simulate sensor noise, and the total variation regularized derivative (33) is used. In this example, the normal form is correctly identified, resulting in accurate limit cycle amplitudes and growth rates (*Bottom Right*). The correct identification of a normal form depends critically on the choice of variables and the nonlinear basis functions used for $\Theta(\mathbf{x})$. In practice, these choices may be informed by machine learning and data mining, by partial knowledge of the physics, and by expert human intuition.

Similarly, time dependence and external forcing or feedback control $\mathbf{u}(t)$ may be added to the vector field:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t), t),$$

$$\dot{t} = 1.$$

Generalizing the SINDy algorithm makes it possible to analyze systems that are externally forced or controlled. For example, the

originate off of the slow manifold, the algorithm incorrectly identifies cubic nonlinearities, and fails to identify the slow manifold.

Normal Forms, Bifurcations, and Parameterized Systems. In practice, many real-world systems depend on parameters, and dramatic changes, or bifurcations, may occur when the parameter is varied. The SINDy algorithm is readily extended to encompass these important parameterized systems, allowing for the discovery of normal forms (31, 50) associated with a bifurcation parameter μ. First, we append μ to the dynamics:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}; \mu), \qquad [9a]$$

$$\dot{\mu} = 0. \qquad [9b]$$

It is then possible to identify $\mathbf{f}(\mathbf{x}; \mu)$ as a sparse combination of functions of $\mathbf{x}$ as well as the bifurcation parameter μ.

Identifying parameterized dynamics is shown in two examples: the 1D logistic map with stochastic forcing,

$$x_{k+1} = \mu x_k (1 - x_k) + \eta_k,$$

and the 2D Hopf normal form (51),

$$\dot{x} = \mu x + \omega y - A x \left(x^2 + y^2\right)$$

$$\dot{y} = -\omega x + \mu y - A y \left(x^2 + y^2\right).$$

The logistic map is a classical model for population dynamics, and the Hopf normal form models spontaneous oscillations in chemical reactions, electrical circuits, and fluid instability.

The noisy measurements and the sparse dynamic reconstructions for both examples are shown in Fig. 3. In the logistic map example, the stochastically forced trajectory is sampled at 10 discrete parameter values, shown in red. From these measurements, the correct parameterized dynamics are identified. The parameterization is accurate enough to capture the cascade of bifurcations as μ is increased, resulting in the detailed bifurcation diagram in Fig. 3. Parameters are identified to within .1% of true values (*SI Appendix*, Appendix C).

and the nonlinear basis functions used for $\Theta(\mathbf{x})$. In practice, these choices may be informed by machine learning and data mining, by partial knowledge of the physics, and by expert human intuition.

Similarly, time dependence and external forcing or feedback control $\mathbf{u}(t)$ may be added to the vector field:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t), t),$$

$$\dot{t} = 1.$$

Generalizing the SINDy algorithm makes it possible to analyze systems that are externally forced or controlled. For example, the climate is both parameterized (50) and has external forcing, including carbon dioxide and solar radiation. The financial market is another important example with forcing and active feedback control.

Discussion

In summary, we have demonstrated a powerful technique to identify nonlinear dynamical systems from data without assumptions on the form of the governing equations. This builds on prior work in symbolic regression but with innovations related to sparse regression, which allow our algorithms to scale to high-dimensional systems. We demonstrate this method on a number of example systems exhibiting chaos, high-dimensional data with low-rank coherence, and parameterized dynamics. As shown in the Lorenz example, the ability to predict a specific trajectory may be less important than the ability to capture the attractor dynamics. The example from fluid dynamics highlights the remarkable ability of this method to extract dynamics in a fluid system that took three decades for experts in the community to explain. There are numerous fields where this method may be applied, where there are ample data and the absence of governing equations, including neuroscience, climate science, epidemiology, and financial markets. Finally, normal forms may be discovered by including parameters in the optimization, as shown in two examples. The identification of sparse governing equations and parameterizations marks a significant step toward the long-held goal of intelligent, unassisted identification of dynamical systems.

We have demonstrated the robustness of the sparse dynamics algorithm to measurement noise and unavailability of derivative

measurements in the Lorenz system (*SI Appendix*, Figs. S6 and S7), logistic map (*SI Appendix*, section 4.4.1), and Hopf normal form (*SI Appendix*, section 4.4.2) examples. In each case, the sparse regression framework appears well-suited to measurement and process noise, especially when derivatives are smoothed using the total-variation regularized derivative.

A significant outstanding issue in the above approach is the correct choice of measurement coordinates and the choice of sparsifying function basis for the dynamics. As shown in *SI Appendix*, Appendix B, the algorithm fails to identify an accurate sparse model when the measurement coordinates and function basis are not amenable to sparse representation. In the successful examples, the coordinates and function spaces were somehow fortunate in that they enabled sparse representation. There is no simple solution to this challenge, and there must be a coordinated effort to incorporate expert knowledge, feature extraction, and other advanced methods. However, in practice, there may be some hope of obtaining the correct coordinate system and function basis without knowing the solution ahead of time, because we often know something about the physics that guide the choice of function space. The failure to identify an accurate sparse model also provides valuable diagnostic information about the coordinates and basis. If we have few measurements, these may be augmented using time-delay coordinates, as demonstrated on the Lorenz system (*SI Appendix*, section 4.5). When there are too many measurements, we may extract coherent structures using dimensionality reduction. We also demonstrate the use of polynomial bases to approximate Taylor series of nonlinear dynamics (*SI Appendix*, Appendix A). The connection between sparse optimization and dynamical systems will hopefully spur developments to automate and improve these choices.

Data science is not a panacea for all problems in science and engineering, but used in the right way, it provides a principled approach to maximally leverage the data that we have and inform what new data to collect. Big data are happening all across the sciences, where the data are inherently dynamic, and where traditional approaches are prone to overfitting. Data discovery algorithms that produce parsimonious models are both rare and desirable. Data science will only become more critical to efforts in science in engineering, such as understanding the neural basis of cognition, extracting and predicting coherent changes in the climate, stabilizing financial markets, managing the spread of disease, and controlling turbulence, where data are abundant, but physical laws remain elusive.

ACKNOWLEDGMENTS. We are grateful for discussions with Bing Brunton and Bernd Noack, and for insightful comments from the referees. We also thank Tony Roberts and Jean-Christophe Loiseau. S.L.B. acknowledges support from the University of Washington. J.L.P. thanks Bill and Melinda Gates for their support of the Institute for Disease Modeling and their sponsorship through the Global Good Fund. J.N.K. acknowledges support from the US Air Force Office of Scientific Research (FA9550-09-0174).

1. Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. *Science* 349(6245):255–260.
2. Marx V (2013) Biology: The big challenges of big data. *Nature* 498(7453):255–260.
3. Bongard J, Lipson H (2007) Automated reverse engineering of nonlinear dynamical systems. *Proc Natl Acad Sci USA* 104(24):9943–9948.
4. Schmidt M, Lipson H (2009) Distilling free-form natural laws from experimental data. *Science* 324(5923):81–85.
5. Koza JR (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA), Vol 1.
6. Crutchfield JP, McNamara BS (1987) Equations of motion from a data series. *Complex Syst* 1(3):417–452.
7. Kevrekidis IG, et al. (2003) Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Commun Math Sci* 1(4):715–762.
8. Sugihara G, et al. (2012) Detecting causality in complex ecosystems. *Science* 338(6106): 496–500.
9. Ye H, et al. (2015) Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proc Natl Acad Sci USA* 112(13):E1569–E1576.
10. Roberts AJ (2014) *Model Emergent Dynamics in Complex Systems* (SIAM, Philadelphia).
27. Brunton SL, Tu JH, Bright I, Kutz JN (2014) Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM J Appl Dyn Syst* 13(4):1716–1732.
28. Proctor JL, Brunton SL, Brunton BW, Kutz JN (2014) Exploiting sparsity and equation-free architectures in complex systems (invited review). *Eur Phys J Spec Top* 223:2665–2684.
29. Bai Z, et al. (2014) Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA J* 53(4):920–930.
30. Arnaldo I, O'Reilly UM, Veeramachaneni K (2015) Building predictive models via feature synthesis. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (ACM, New York), pp 983–990.
31. Holmes P, Guckenheimer J (1983) Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. *Applied Mathematical Sciences* (Springer, Berlin), Vol 42.
32. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Physica D* 60(1):259–268.
33. Chartrand R (2011) Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics* 2011(2011):164564.
34. Gavish M, Donoho DL (2014) The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Trans Inf Theory* 60(8):5040–5053.
35. Holmes PJ, Lumley JL, Berkooz G, Rowley CW (2012) *Turbulence, Coherent Structures,*

systems. *Proc Natl Acad Sci USA* 104(24):9943–9948.
4. Schmidt M, Lipson H (2009) Distilling free-form natural laws from experimental data. *Science* 324(5923):81–85.
5. Koza JR (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA), Vol 1.
6. Crutchfield JP, McNamara BS (1987) Equations of motion from a data series. *Complex Syst* 1(3):417–452.
7. Kevrekidis IG, et al. (2003) Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Commun Math Sci* 1(4):715–762.
8. Sugihara G, et al. (2012) Detecting causality in complex ecosystems. *Science* 338(6106): 496–500.
9. Ye H, et al. (2015) Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proc Natl Acad Sci USA* 112(13):E1569–E1576.
10. Roberts AJ (2014) *Model Emergent Dynamics in Complex Systems* (SIAM, Philadelphia).
11. Schmidt MD, et al. (2011) Automated refinement and inference of analytical models for metabolic networks. *Phys Biol* 8(5):055011.
12. Daniels BC, Nemenman I (2015) Automated adaptive inference of phenomenological dynamical models. *Nat Commun* 6:8133.
13. Daniels BC, Nemenman I (2015) Efficient inference of parsimonious phenomenological models of cellular dynamics using S-systems and alternating regression. *PLoS One* 10(3):e0119821.
14. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J R Stat Soc, B* 58(1):267–288.
15. Hastie T, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning* (Springer, New York), Vol 2.
16. James G, Witten D, Hastie T, Tibshirani R (2013) *An Introduction to Statistical Learning* (Springer, New York).
17. Donoho DL (2006) Compressed sensing. *IEEE Trans Inf Theory* 52(4):1289–1306.
18. Candès EJ, Romberg J, Tao T (2006) Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans Inf Theory* 52(2):489–509.
19. Candès EJ, Romberg J, Tao T (2006) Stable signal recovery from incomplete and inaccurate measurements. *Commun Pure Appl Math* 59(8):1207–1223.
20. Candès EJ, Wakin MB (2008) An introduction to compressive sampling. *IEEE Signal Processing Magazine* 25(2):21–30.
21. Baraniuk RG (2007) Compressive sensing. *IEEE Signal Process Mag* 24(4):118–120.
22. Tropp JA, Gilbert AC (2007) Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans Inf Theory* 53(12):4655–4666.
23. Wang WX, Yang R, Lai YC, Kovanis V, Grebogi C (2011) Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Phys Rev Lett* 106(15):154101.
24. Schaeffer H, Caflisch R, Hauck CD, Osher S (2013) Sparse dynamics for partial differential equations. *Proc Natl Acad Sci USA* 110(17):6634–6639.
25. Ozoliņš V, Lai R, Caflisch R, Osher S (2013) Compressed modes for variational problems in mathematics and physics. *Proc Natl Acad Sci USA* 110(46):18368–18373.
26. Mackey A, Schaeffer H, Osher S (2014) On the compressive spectral method. *Multiscale Model Simul* 12(4):1800–1827.
29. Bai Z, et al. (2014) Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA J* 53(4):920–930.
30. Arnaldo I, O'Reilly UM, Veeramachaneni K (2015) Building predictive models via feature synthesis. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (ACM, New York), pp 983–990.
31. Holmes P, Guckenheimer J (1983) Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. *Applied Mathematical Sciences* (Springer, Berlin), Vol 42.
32. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Physica D* 60(1):259–268.
33. Chartrand R (2011) Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics* 2011(2011):164564.
34. Gavish M, Donoho DL (2014) The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Trans Inf Theory* 60(8):5040–5053.
35. Holmes PJ, Lumley JL, Berkooz G, Rowley CW (2012) *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge Monographs in Mechanics (Cambridge Univ Press, Cambridge, UK), 2nd Ed.
36. Majda AJ, Harlim J (2007) Information flow between subspaces of complex dynamical systems. *Proc Natl Acad Sci USA* 104(23):9558–9563.
37. Berkooz G, Holmes P, Lumley JL (1993) The proper orthogonal decomposition in the analysis of turbulent flows. *Annu Rev Fluid Mech* 23:539–575.
38. Takens F (1981) Detecting strange attractors in turbulence. *Lect Notes Math* 898:366–381.
39. Marsden JE, Ratiu TS (1999) *Introduction to Mechanics and Symmetry* (Springer, New York), 2nd Ed.
40. Lorenz EN (1963) Deterministic nonperiodic flow. *J Atmos Sci* 20:130–141.
41. Rowley CW, Mezić I, Bagheri S, Schlatter P, Henningson D (2009) Spectral analysis of nonlinear flows. *J Fluid Mech* 645:115–127.
42. Schmid PJ (2010) Dynamic mode decomposition of numerical and experimental data. *J Fluid Mech* 656:5–28.
43. Mezic I (2013) Analysis of fluid flows via spectral properties of the Koopman operator. *Annu Rev Fluid Mech* 45:357–378.
44. Taira K, Colonius T (2007) The immersed boundary method: A projection approach. *J Comput Phys* 225(2):2118–2137.
45. Colonius T, Taira K (2008) A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Comput Methods Appl Mech Eng* 197:2131–2146.
46. Ruelle D, Takens F (1971) On the nature of turbulence. *Commun Math Phys* 20:167–192.
47. Jackson CP (1987) A finite-element study of the onset of vortex shedding in flow past variously shaped bodies. *J Fluid Mech* 182:23–45.
48. Zebib Z (1987) Stability of viscous flow past a circular cylinder. *J Eng Math* 21:155–165.
49. Noack BR, Afanasiev K, Morzynski M, Tadmor G, Thiele F (2003) A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *J Fluid Mech* 497:335–363.
50. Majda AJ, Franzke C, Crommelin D (2009) Normal forms for reduced stochastic climate models. *Proc Natl Acad Sci USA* 106(10):3649–3653.
51. Marsden JE, McCracken M (1976) *The Hopf Bifurcation and Its Applications* (Springer, New York), Vol 19.

Programming Techniques
S.L. Graham, R.L. Rivest* Editors

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:
(1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
(2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, *n*, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e * d \equiv 1 (\mathrm{mod}\ (p-1) * (q-1))$. The

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

$$D(E(M)) = M. \tag{1}$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e * d \equiv 1 (\mathrm{mod}\ (p-1) * (q-1))$. The security of the system rests in part on the difficulty of factoring the published divisor, n.

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

* Note. This paper was submitted prior to the time that Rivest became editor of the department, and editorial consideration was completed under the former editor, G. K. Manacher.

Authors' Address: MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

$$D(E(M)) = M. \quad (1)$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

$$E(D(M)) = M. \quad (2)$$

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext* C. Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a very *inefficient* method of computing D(C): testing all possible messages M until one such that E(M) = C is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)–(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but

did not present any examples. These functions are called "one-way" because they are easy to compute in one direction but (apparently) very difficult to compute in the other direction. They are called "trap-door" functions since the inverse functions are in fact easy to compute once certain private "trap-door" information is known. A trap-door one-way function which also satisfies (d) must be a permutation: every message is the ciphertext for some other message and every ciphertext is itself a permissible message. (The mapping is "one-to-one" and "onto"). Property (d) is needed only to implement "signatures".

The reader is encouraged to read Diffie and Hellman's excellent article [1] for further background, for elaboration of the concept of a public-key cryptosystem, and for a discussion of other problems in the area of cryptography. The ways in which a public-key cryptosystem can ensure privacy and enable "signatures" (described in Sections III and IV below) are also due to Diffie and Hellman.

For our scenarios we suppose that A and B (also known as Alice and Bob) are two users of a public-key cryptosystem. We will distinguish their encryption and decryption procedures with subscripts: E_A, D_A, E_B, D_B.

III. Privacy

Encryption is the standard means of rendering a communication private. The sender enciphers each message before transmitting it to the receiver. The receiver (but no unauthorized person) knows the appropriate deciphering function to apply to the received message to obtain the original message. An eavesdropper who hears the transmitted message hears only "garbage" (the ciphertext) which makes no sense to him since he does not know how to decrypt it.

The large volume of personal and sensitive infor-

a public-key cryptosystem? First, he retrieves E_A from the public file. Then he sends her the enciphered message $E_A(M)$. Alice deciphers the message by computing $D_A(E_A(M)) = M$. By property (c) of the public-key cryptosystem only she can decipher $E_A(M)$. She can encipher a private response with E_B, also available in the public file.

Observe that no private transactions between Alice and Bob are needed to establish private communication. The only "setup" required is that each user who wishes to receive private communications must place his enciphering algorithm in the public file.

Two users can also establish private communication over an insecure communications channel without consulting a public file. Each user sends his encryption key to the other. Afterwards all messages are enciphered with the encryption key of the recipient, as in the public-key system. An intruder listening in on the channel cannot decipher any messages, since it is not possible to derive the decryption keys from the encryption keys. (We assume that the intruder cannot modify or insert messages into the channel.) Ralph Merkle has developed another solution [5] to this problem.

A public-key cryptosystem can be used to "bootstrap" into a standard encryption scheme such as the NBS method. Once secure communications have been established, the first message transmitted can be a key to use in the NBS scheme to encode all following messages. This may be desirable if encryption with our method is slower than with the standard scheme. (The NBS scheme is probably somewhat faster if special-purpose hardware encryption devices are used; our scheme may be faster on a general-purpose computer since multiprecision arithmetic operations are simpler to implement than complicated bit manipulations.)

III. Privacy

Encryption is the standard means of rendering a communication private. The sender enciphers each message before transmitting it to the receiver. The receiver (but no unauthorized person) knows the appropriate deciphering function to apply to the received message to obtain the original message. An eavesdropper who hears the transmitted message hears only "garbage" (the ciphertext) which makes no sense to him since he does not know how to decrypt it.

The large volume of personal and sensitive information currently held in computerized data banks and transmitted over telephone lines makes encryption increasingly important. In recognition of the fact that efficient, high-quality encryption techniques are very much needed but are in short supply, the National Bureau of Standards has recently adopted a "Data Encryption Standard" [13, 14], developed at IBM. The new standard does not have property (c), needed to implement a public-key cryptosystem.

All classical encryption methods (including the NBS standard) suffer from the "key distribution problem." The problem is that before a private communication can begin, *another* private transaction is necessary to distribute corresponding encryption and decryption keys to the sender and receiver, respectively. Typically a private courier is used to carry a key from the sender to the receiver. Such a practice is not feasible if an electronic mail system is to be rapid and inexpensive. A public-key cryptosystem needs no private couriers; the keys can be distributed over the insecure communications channel.

How can Bob send a private message M to Alice in strap" into a standard encryption scheme such as the NBS method. Once secure communications have been established, the first message transmitted can be a key to use in the NBS scheme to encode all following messages. This may be desirable if encryption with our method is slower than with the standard scheme. (The NBS scheme is probably somewhat faster if special-purpose hardware encryption devices are used; our scheme may be faster on a general-purpose computer since multiprecision arithmetic operations are simpler to implement than complicated bit manipulations.)

IV. Signatures

If electronic mail systems are to replace the existing paper mail system for business transactions, "signing" an electronic message must be possible. The recipient of a signed message has proof that the message originated from the sender. This quality is stronger than mere authentication (where the recipient can verify that the message came from the sender); the recipient can convince a "judge" that the signer sent the message. To do so, he must convince the judge that he did not forge the signed message himself! In an authentication problem the recipient does not worry about this possibility, since he only wants to satisfy *himself* that the message came from the sender.

An electronic signature must be *message*-dependent, as well as *signer*-dependent. Otherwise the recipient could modify the message before showing the message-signature pair to a judge. Or he could attach the signature to any message whatsoever, since it is impossible to detect electronic "cutting and pasting."

To implement signatures the public-key cryptosys-

tem must be implemented with trap-door one-way permutations (i.e. have property (d)), since the decryption algorithm will be applied to unenciphered messages.

How can user Bob send Alice a "signed" message M in a public-key cryptosystem? He first computes his "signature" S for the message M using D_B:

$$S = D_B(M).$$

(Deciphering an unenciphered message "makes sense" by property (d) of a public key cryptosystem: each message is the ciphertext for some other message.) He then encrypts S using E_A (for privacy), and sends the result $E_A(S)$ to Alice. He need not send M as well; it can be computed from S.

Alice first decrypts the ciphertext with D_A to obtain S. She knows who is the presumed sender of the signature (in this case, Bob); this can be given if necessary in plain text attached to S. She then extracts the message with the encryption procedure of the sender, in this case E_B (available on the public file):

$$M = E_B(S).$$

She now possesses a message-signature pair (M, S) with properties similar to those of a signed paper document.

Bob cannot later deny having sent Alice this message, since no one else could have created $S = D_B(M)$. Alice can convince a "judge" that $E_B(S) = M$, so she has proof that Bob signed the document.

Clearly Alice cannot modify M to a different version M′, since then she would have to create the corresponding signature $S' = D_B(M')$ as well.

Therefore Alice has received a message "signed" by Bob, which she can "prove" that he sent, but which she cannot modify. (Nor can she forge his signature for any other message.)

messages purporting to be from the public file. The user would like to be sure that he actually obtains the encryption procedure of his desired correspondent and not, say, the encryption procedure of the intruder. This danger disappears if the public file "signs" each message it sends to a user. The user can check the signature with the public file's encryption algorithm E_{PF}. The problem of "looking up" E_{PF} itself in the public file is avoided by giving each user a description of E_{PF} when he first shows up (in person) to join the public-key cryptosystem and to deposit his public encryption procedure. He then stores this description rather than ever looking it up again. The need for a courier between every pair of users has thus been replaced by the requirement for a single secure meeting between each user and the public-file manager when the user joins the system. Another solution is to give each user, when he signs up, a book (like a telephone directory) containing all the encryption keys of users in the system.

V. Our Encryption and Decryption Methods

To encrypt a message M with our method, using a public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n - 1$. (Break a long message into a series of blocks, and represent each block as such an integer.) Use any standard representation. The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the ciphertext C) is the remainder when M^e is divided by n.

To decrypt the ciphertext, raise it to another power d, again modulo n. The encryption and decryp-

with properties similar to those of a signed paper document.

Bob cannot later deny having sent Alice this message, since no one else could have created $S = D_B(M)$. Alice can convince a "judge" that $E_B(S) = M$, so she has proof that Bob signed the document.

Clearly Alice cannot modify M to a different version M′, since then she would have to create the corresponding signature $S' = D_B(M')$ as well.

Therefore Alice has received a message "signed" by Bob, which she can "prove" that he sent, but which she cannot modify. (Nor can she forge his signature for any other message.)

An electronic checking system could be based on a signature system such as the above. It is easy to imagine an encryption device in your home terminal allowing you to sign checks that get sent by electronic mail to the payee. It would only be necessary to include a unique check number in each check so that even if the payee copies the check the bank will only honor the first version it sees.

Another possibility arises if encryption devices can be made fast enough: it will be possible to have a telephone conversation in which every word spoken is signed by the encryption device before transmission.

When encryption is used for signatures as above, it is important that the encryption device not be "wired in" between the terminal (or computer) and the communications channel, since a message may have to be successively enciphered with several keys. It is perhaps more natural to view the encryption device as a "hardware subroutine" that can be executed as needed.

We have assumed above that each user can always access the public file reliably. In a "computer network" this might be difficult; an "intruder" might forge

public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n - 1$. (Break a long message into a series of blocks, and represent each block as such an integer.) Use any standard representation. The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the ciphertext C) is the remainder when M^e is divided by n.

To decrypt the ciphertext, raise it to another power d, again modulo n. The encryption and decryption algorithms E and D are thus:

$$C \equiv E(M) \equiv M^e \pmod{n}, \text{ for a message M.}$$
$$D(C) \equiv C^d \pmod{n}, \text{ for a ciphertext C.}$$

Note that encryption does not increase the size of a message; both the message and the ciphertext are integers in the range 0 to $n - 1$.

The *encryption key* is thus the pair of positive integers (e, n). Similarly, the *decryption key* is the pair of positive integers (d, n). Each user makes his encryption key public, and keeps the corresponding decryption key private. (These integers should properly be subscripted as in n_A, e_A, and d_A, since each user has his own set. However, we will only consider a typical set, and will omit the subscripts.)

How should you choose your encryption and decryption keys, if you want to use our method?

You first compute n as the product of two primes p and q:

$$n = p * q.$$

These primes are very large, "random" primes. Al-

though you will make n public, the factors p and q will be effectively hidden from everyone else due to the enormous difficulty of factoring n. This also hides the way d can be derived from e.

You then pick the integer d to be a large, random integer which is relatively prime to $(p - 1) * (q - 1)$. That is, check that d satisfies:

$\gcd(d, (p - 1) * (q - 1)) = 1$
("gcd" means "greatest common divisor").

The integer e is finally computed from p, q, and d to be the "multiplicative inverse" of d, modulo $(p - 1) * (q - 1)$. Thus we have

$$e * d \equiv 1 \pmod{(p - 1) * (q - 1)}.$$

We prove in the next section that this guarantees that (1) and (2) hold, i.e. that E and D are inverse permutations. Section VII shows how each of the above operations can be done efficiently.

The aforementioned method should not be confused with the "exponentiation" technique presented by Diffie and Hellman [1] to solve the key distribution problem. Their technique permits two users to determine a key in common to be used in a normal cryptographic system. It is not based on a trap-door one-way permutation. Pohlig and Hellman [8] study a scheme related to ours, where exponentiation is done modulo a prime number.

VI. The Underlying Mathematics

We demonstrate the correctness of the deciphering algorithm using an identity due to Euler and Fermat [7]: for any integer (message) M which is relatively prime to n,

$$M^{\varphi(n)} \equiv 1 \pmod{n}. \qquad (3)$$

Here $\varphi(n)$ is the Euler totient function giving the

$$M^{e*d} \equiv M^{k*\varphi(n)+1} \pmod{n} \qquad \text{(for some integer } k\text{)}.$$

From (3) we see that for all M such that p does not divide M

$$M^{p-1} \equiv 1 \pmod{p}$$

and since $(p - 1)$ divides $\varphi(n)$

$$M^{k*\varphi(n)+1} \equiv M \pmod{p}.$$

This is trivially true when $M \equiv 0 \pmod{p}$, so that this equality actually holds for *all* M. Arguing similarly for q yields

$$M^{k*\varphi(n)+1} \equiv M \pmod{q}.$$

Together these last two equations imply that for all M,

$$M^{e*d} \equiv M^{k*\varphi(n)+1} \equiv M \pmod{n}.$$

This implies (1) and (2) for all M, $0 \leq M < n$. Therefore E and D are inverse permutations. (We thank Rich Schroeppel for suggesting the above improved version of the authors' previous proof.)

VII. Algorithms

To show that our method is practical, we describe an efficient algorithm for each required operation.

A. How to Encrypt and Decrypt Efficiently

Computing $M^e \pmod{n}$ requires at most $2 * \log_2(e)$ multiplications and $2 * \log_2(e)$ divisions using the following procedure (decryption can be performed similarly using d instead of e):

Step 1. Let $e_k e_{k-1} \ldots e_1 e_0$ be the binary representation of e.
Step 2. Set the variable C to 1.
Step 3. Repeat steps 3a and 3b for $i = k, k - 1, \ldots, 0$:

graphic system. It is not based on a trap-door one-way permutation. Pohlig and Hellman [8] study a scheme related to ours, where exponentiation is done modulo a prime number.

VI. The Underlying Mathematics

We demonstrate the correctness of the deciphering algorithm using an identity due to Euler and Fermat [7]: for any integer (message) M which is relatively prime to n,

$$M^{\varphi(n)} \equiv 1 \pmod{n}. \tag{3}$$

Here $\varphi(n)$ is the Euler totient function giving the number of positive integers less than n which are relatively prime to n. For prime numbers p,

$$\varphi(p) = p - 1.$$

In our case, we have by elementary properties of the totient function [7]:

$$\begin{aligned} \varphi(n) &= \varphi(p) * \varphi(q), \\ &= (p-1) * (q-1) \\ &= n - (p+q) + 1. \end{aligned} \tag{4}$$

Since d is relatively prime to $\varphi(n)$, it has a multiplicative inverse e in the ring of integers modulo $\varphi(n)$:

$$e * d \equiv 1 \pmod{\varphi(n)}. \tag{5}$$

We now prove that equations (1) and (2) hold (that is, that deciphering works correctly if e and d are chosen as above). Now

$$D(E(M)) \equiv (E(M))^d \equiv (M^e)^d \equiv M^{e*d} \pmod{n}$$

$$E(D(M)) \equiv (D(M))^e \equiv (M^d)^e \equiv M^{e*d} \pmod{n}$$

and

an efficient algorithm for each required operation.

A. How to Encrypt and Decrypt Efficiently

Computing $M^e \pmod{n}$ requires at most $2 * \log_2(e)$ multiplications and $2 * \log_2(e)$ divisions using the following procedure (decryption can be performed similarly using d instead of e):

Step 1. Let $e_k e_{k-1} \ldots e_1 e_0$ be the binary representation of e.
Step 2. Set the variable C to 1.
Step 3. Repeat steps 3a and 3b for $i = k, k-1, \ldots, 0$:
  Step 3a. Set C to the remainder of C^2 when divided by n.
  Step 3b. If $e_i = 1$, then set C to the remainder of $C * M$ when divided by n.
Step 4. Halt. Now C is the encrypted form of M.

This procedure is called "exponentiation by repeated squaring and multiplication." This procedure is half as good as the best; more efficient procedures are known. Knuth [3] studies this problem in detail.

The fact that the enciphering and deciphering are identical leads to a simple implementation. (The whole operation can be implemented on a few special-purpose integrated circuit chips.)

A high-speed computer can encrypt a 200-digit message M in a few seconds; special-purpose hardware would be much faster. The encryption time per block increases no faster than the cube of the number of digits in n.

B. How to Find Large Prime Numbers

Each user must (privately) choose two large ran-

dom prime numbers p and q to create his own encryption and decryption keys. These numbers must be large so that it is not computationally feasible for anyone to factor $n = p * q$. (Remember that n, but not p or q, will be in the public file.) We recommend using 100-digit (decimal) prime numbers p and q, so that n has 200 digits.

To find a 100-digit "random" prime number, generate (odd) 100-digit random numbers until a prime number is found. By the prime number theorem [7], about $(\ln 10^{100})/2 = 115$ numbers will be tested before a prime is found.

To test a large number b for primality we recommend the elegant "probabilistic" algorithm due to Solovay and Strassen [12]. It picks a random number a from a uniform distribution on $\{1, \ldots, b-1\}$, and tests whether

$$\gcd(a, b) = 1 \text{ and } J(a, b) \equiv a^{(b-1)/2} (\text{mod } b), \tag{6}$$

where $J(a, b)$ is the Jacobi symbol [7]. If b is prime (6) is always true. If b is composite (6) will be false with probability at least $1/2$. If (6) holds for 100 randomly chosen values of a then b is almost certainly prime; there is a (negligible) chance of one in 2^{100} that b is composite. Even if a composite were accidentally used in our system, the receiver would probably detect this by noticing that decryption didn't work correctly. When b is odd, $a \leq b$, and $\gcd(a, b) = 1$, the Jacobi symbol $J(a, b)$ has a value in $\{-1, 1\}$ and can be efficiently computed by the program:

$J(a, b) =$ **if** $a = 1$ **then** 1 **else**
if a is even **then** $J(a/2, b) * (-1)^{(b^2-1)/8}$
else $J(b(\text{mod } a), a) * (-1)^{(a-1)*(b-1)/4}$

(The computations of $J(a, b)$ and $\gcd(a, b)$ can be nicely combined, too.) Note that this algorithm does using the factorization of $p - 1$. We omit a discussion of this since the probabilistic method is adequate.

C. How to Choose d

It is very easy to choose a number d which is relatively prime to $\varphi(n)$. For example, any prime number greater than $\max(p, q)$ will do. It is important that d should be chosen from a large enough set so that a cryptanalyst cannot find it by direct search.

D. How to Compute e from d and $\varphi(n)$

To compute e, use the following variation of Euclid's algorithm for computing the greatest common divisor of $\varphi(n)$ and d. (See exercise 4.5.2.15 in [3].) Calculate $\gcd(\varphi(n), d)$ by computing a series $x_0, x_1, x_2, \ldots$, where $x_0 = \varphi(n)$, $x_1 = d$, and $x_{i+1} \equiv x_{i-1} (\text{mod } x_i)$, until an x_k equal to 0 is found. Then $\gcd(x_0, x_1) = x_{k-1}$. Compute for each x_i numbers a_i and b_i such that $x_i = a_i * x_0 + b_i * x_1$. If $x_{k-1} = 1$ then b_{k-1} is the multiplicative inverse of $x_1 \pmod{x_0}$. Since k will be less than $2 * \log_2(n)$, this computation is very rapid.

If e turns out to be less than $\log_2(n)$, start over by choosing another value of d. This guarantees that every encrypted message (except $M = 0$ or $M = 1$) undergoes some "wrap-around" (reduction modulo n).

VIII. A Small Example

Consider the case $p = 47, q = 59, n = p * q = 47 * 59 = 2773$, and $d = 157$. Then $\varphi(2773) = 46 * 58 = 2668$, and e can be computed as follows:

$x_0 = 2668,$	$a_0 = 1,$	$b_0 = 0,$
$x_1 = 157,$	$a_1 = 0,$	$b_1 = 1,$
$x_2 = 156,$	$a_2 = 1,$	$b_2 = -16$ (since $2668 = 157 * 16 + 156$),
$x_3 = 1,$	$a_3 = -1,$	$b_3 = 17$ (since $157 = 1 * 156 + 1$)

composite. Even if a composite were accidentally used in our system, the receiver would probably detect this by noticing that decryption didn't work correctly. When b is odd, $a \leq b$, and $\gcd(a, b) = 1$, the Jacobi symbol $J(a, b)$ has a value in $\{-1, 1\}$ and can be efficiently computed by the program:

$J(a, b) =$ **if** $a = 1$ **then** 1 **else**
if a is even **then** $J(a/2, b) * (-1)^{(b^2-1)/8}$
else $J(b \pmod a, a) * (-1)^{(a-1)*(b-1)/4}$

(The computations of $J(a, b)$ and $\gcd(a, b)$ can be nicely combined, too.) Note that this algorithm does *not* test a number for primality by trying to factor it. Other efficient procedures for testing a large number for primality are given in [6, 9, 11].

To gain additional protection against sophisticated factoring algorithms, p and q should differ in length by a few digits, both $(p - 1)$ and $(q - 1)$ should contain large prime factors, and $\gcd(p - 1, q - 1)$ should be small. The latter condition is easily checked.

To find a prime number p such that $(p - 1)$ has a large prime factor, generate a large random prime number u, then let p be the first prime in the sequence $i * u + 1$, for $i = 2, 4, 6, \ldots$. (This shouldn't take too long.) Additional security is provided by ensuring that $(u - 1)$ also has a large prime factor.

A high-speed computer can determine in several seconds whether a 100-digit number is prime, and can find the first prime after a given point in a minute or two.

Another approach to finding large prime numbers is to take a number of known factorization, add one to it, and test the result for primality. If a prime p is found it is possible to *prove* that it really is prime by

every encrypted message (except M = 0 or M = 1) undergoes some "wrap-around" (reduction modulo n).

VIII. A Small Example

Consider the case $p = 47, q = 59, n = p * q = 47 * 59 = 2773$, and $d = 157$. Then $\varphi(2773) = 46 * 58 = 2668$, and e can be computed as follows:

$x_0 = 2668,$	$a_0 = 1,$	$b_0 = 0,$
$x_1 = 157,$	$a_1 = 0,$	$b_1 = 1,$
$x_2 = 156,$	$a_2 = 1,$	$b_2 = -16$ (since $2668 = 157 * 16 + 156$),
$x_3 = 1,$	$a_3 = -1,$	$b_3 = 17$ (since $157 = 1 * 156 + 1$).

Therefore $e = 17$, the multiplicative inverse (mod 2668) of $d = 157$.

With $n = 2773$ we can encode two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02, . . . , Z = 26. Thus the message

ITS ALL GREEK TO ME

(Julius Caesar, I, ii, 288, paraphrased) is encoded:

0920 1900 0112 1200 0718
0505 1100 2015 0013 0500

Since $e = 10001$ in binary, the first block (M = 920) is enciphered:

$M^{17} \equiv (((((1)^2 * M)^2)^2)^2)^2 * M \equiv 948 \pmod{2773}$.

The whole message is enciphered as:

0948 2342 1084 1444 2663
2390 0778 0774 0219 1655.

The reader can check that deciphering works: $948^{157} \equiv 920 \pmod{2773}$, etc.

IX. Security of the Method: Cryptanalytic Approaches

Since no techniques exist to *prove* that an encryption scheme is secure, the only test available is to see whether anyone can think of a way to break it. The NBS standard was "certified" this way; seventeen man-years at IBM were spent fruitlessly trying to break that scheme. Once a method has successfully resisted such a concerted attack it may for practical purposes be considered secure. (Actually there is some controversy concerning the security of the NBS method [2].)

We show in the next sections that all the obvious approaches for breaking our system are at least as difficult as factoring n. While factoring large numbers is not provably difficult, it is a well-known problem that has been worked on for the last three hundred years by many famous mathematicians. Fermat (1601?–1665) and Legendre (1752–1833) developed factoring algorithms; some of today's more efficient algorithms are based on the work of Legendre. As we shall see in the next section, however, no one has yet found an algorithm which can factor a 200-digit number in a reasonable amount of time. We conclude that our system has already been partially "certified" by these previous efforts to find efficient factoring algorithms.

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to *generate* the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could

Table I.

Digits	Number of operations	Time
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

factor n with Schroeppel's method, and the time required if each operation uses one microsecond, for various lengths of the number n (in decimal digits):

We recommend that n be about 200 digits long. Longer or shorter lengths can be used depending on the relative importance of encryption speed and security in the application at hand. An 80-digit n provides moderate security against an attack using current technology; using 200 digits provides a margin of safety against future developments. This flexibility to choose a key-length (and thus a level of security) to suit a particular application is a feature not found in many of the previous encryption schemes (such as the NBS scheme).

B. Computing $\varphi(n)$ Without Factoring n

If a cryptanalyst could compute $\varphi(n)$ then he could break the system by computing d as the multiplicative inverse of e modulo $\varphi(n)$ (using the procedure of Section VII D).

We argue that this approach is no easier than factoring n since it enables the cryptanalyst to easily factor n using $\varphi(n)$. This approach to factoring n has not turned out to be practical.

How can n be factored using $\varphi(n)$? First, $(p + q)$ is obtained from n and $\varphi(n) = n - (p + q) + 1$. Then $(p - q)$ is the square root of $(p + q)^2 - 4n$. Finally, q is half the difference of $(p + q)$ and $(p - q)$.

these previous efforts to find efficient factoring algorithms.

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to *generate* the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could erase the decryption key if it was tampered with.)

A. Factoring n

Factoring n would enable an enemy cryptanalyst to "break" our method. The factors of n enable him to compute $\varphi(n)$ and thus d. Fortunately, factoring a number seems to be much more difficult than determining whether it is prime or composite.

A large number of factoring algorithms exist. Knuth [3, Section 4.5.4] gives an excellent presentation of many of them. Pollard [9] presents an algorithm which factors a number n in time $O(n^{1/4})$.

The fastest factoring algorithm known to the authors is due to Richard Schroeppel (unpublished); it can factor n in approximately

$$\begin{aligned} &\exp(\mathrm{sqrt}(\ln(n) * \ln(\ln(n)))) \\ &= n^{\mathrm{sqrt}(\ln(\ln(n))/\ln(n))} \\ &= (\ln(n))^{\mathrm{sqrt}(\ln(n)/\ln(\ln(n)))} \end{aligned}$$

steps (here ln denotes the natural logarithm function). Table I gives the number of operations needed to

B. Computing $\varphi(n)$ Without Factoring n

If a cryptanalyst could compute $\varphi(n)$ then he could break the system by computing d as the multiplicative inverse of e modulo $\varphi(n)$ (using the procedure of Section VII D).

We argue that this approach is no easier than factoring n since it enables the cryptanalyst to easily factor n using $\varphi(n)$. This approach to factoring n has not turned out to be practical.

How can n be factored using $\varphi(n)$? First, $(p + q)$ is obtained from n and $\varphi(n) = n - (p + q) + 1$. Then $(p - q)$ is the square root of $(p + q)^2 - 4n$. Finally, q is half the difference of $(p + q)$ and $(p - q)$.

Therefore breaking our system by computing $\varphi(n)$ is no easier than breaking our system by factoring n. (This is why n must be composite; $\varphi(n)$ is trivial to compute if n is prime.)

C. Determining d Without Factoring n or Computing $\varphi(n)$.

Of course, d should be chosen from a large enough set so that a direct search for it is unfeasible.

We argue that computing d is no easier for a cryptanalyst than factoring n, since once d is known n could be factored easily. This approach to factoring has also not turned out to be fruitful.

A knowledge of d enables n to be factored as follows. Once a cryptanalyst knows d he can calculate $e * d - 1$, which is a multiple of $\varphi(n)$. Miller [6] has shown that n can be factored using any multiple of $\varphi(n)$. Therefore if n is large a cryptanalyst should not be able to determine d any easier than he can factor n.

A cryptanalyst may hope to find a d' which is equivalent to the d secretly held by a user of the

public-key cryptosystem. If such values d' were common then a brute-force search could break the system. However, all such d' differ by the least common multiple of $(p-1)$ and $(q-1)$, and finding one enables n to be factored. (In (3) and (5), $\varphi(n)$ can be replaced by $\text{lcm}(p-1, q-1)$.) Finding any such d' is therefore as difficult as factoring n.

D. Computing D in Some Other Way

Although this problem of "computing eth roots modulo n without factoring n" is not a well-known difficult problem like factoring, we feel reasonably confident that it is computationally intractable. It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm. This would establish that any way of breaking our scheme must be as difficult as factoring. We have not been able to prove this conjecture, however.

Our method should be certified by having the above conjecture of intractability withstand a concerted attempt to disprove it. The reader is challenged to find a way to "break" our method.

X. Avoiding "Reblocking" when Encrypting a Signed Message

A signed message may have to be "reblocked" for encryption since the signature n may be larger than the encryption n (every user has his own n). This can be avoided as follows. A threshold value h is chosen (say $h = 10^{199}$) for the public-key cryptosystem. Every user maintains *two* public (e, n) pairs, one for enciphering and one for signature-verification, where every signature n is less than h, and every enciphering n is greater than h. Reblocking to encipher a signed message is then unnecessary; the message is blocked ac-

couriers to carry keys, and it also permits one to "sign" digitized documents.

The security of this system needs to be examined in more detail. In particular, the difficulty of factoring large numbers should be examined very closely. The reader is urged to find a way to "break" the system. Once the method has withstood all attacks for a sufficient length of time it may be used with a reasonable amount of confidence.

Our encryption function is the only candidate for a "trap-door one-way permutation" known to the authors. It might be desirable to find other examples, to provide alternative implementations should the security of our system turn out someday to be inadequate. There are surely also many new applications to be discovered for these functions.

Acknowledgments. We thank Martin Hellman, Richard Schroeppel, Abraham Lempel, and Roger Needham for helpful discussions, and Wendy Glasser for her assistance in preparing the initial manuscript. Xerox PARC provided support and some marvelous text-editing facilities for preparing the final manuscript.

Received April 4, 1977; revised September 1, 1977

References

1. Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22*, 6 (Nov. 1976), 644–654.
2. Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer 10* (June 1977), 74–84.
3. Knuth, D. E. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1969.
4. Levine, J., and Brawley, J.V. Some cryptographic applications of permutation polynomials. *Cryptologia 1* (Jan. 1977), 76–92.
5. Merkle, R. Secure communications over an insecure channel. Submitted to *Comm. ACM.*
6. Miller, G.L. Riemann's hypothesis and tests for primality. Proc. Seventh Annual ACM Symp. on the Theory of Comptng.

Message

A signed message may have to be "reblocked" for encryption since the signature n may be larger than the encryption n (every user has his own n). This can be avoided as follows. A threshold value h is chosen (say $h = 10^{199}$) for the public-key cryptosystem. Every user maintains *two* public (e, n) pairs, one for enciphering and one for signature-verification, where every signature n is less than h, and every enciphering n is greater than h. Reblocking to encipher a signed message is then unnecessary; the message is blocked according to the transmitter's signature n.

Another solution uses a technique given in [4]. Each user has a single (e, n) pair where n is between h and $2h$, where h is a threshold as above. A message is encoded as a number less than h and enciphered as before, except that if the ciphertext is greater than h, it is repeatedly re-enciphered until it is less than h. Similarly for decryption the ciphertext is repeatedly deciphered to obtain a value less than h. If n is near h re-enciphering will be infrequent. (Infinite looping is not possible, since at worst a message is enciphered as itself.)

XI. Conclusions

We have proposed a method for implementing a public-key cryptosystem whose security rests in part on the difficulty of factoring large numbers. If the security of our method proves to be adequate, it permits secure communications to be established without the use of

Received April 4, 1977; revised September 1, 1977

References

1. Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22*, 6 (Nov. 1976), 644–654.
2. Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer 10* (June 1977), 74–84.
3. Knuth, D. E. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms*. Addison-Wesley, Reading, Mass., 1969.
4. Levine, J., and Brawley, J.V. Some cryptographic applications of permutation polynomials. *Cryptologia 1* (Jan. 1977), 76–92.
5. Merkle, R. Secure communications over an insecure channel. Submitted to *Comm. ACM*.
6. Miller, G.L. Riemann's hypothesis and tests for primality. Proc. Seventh Annual ACM Symp. on the Theory of Comptng. Albuquerque, New Mex., May 1975, pp. 234–239; extended vers. available as Res. Rep. CS-75-27, Dept. of Comptr. Sci., U. of Waterloo, Waterloo, Ont., Canada, Oct. 1975.
7. Niven, I., and Zuckerman, H.S. *An Introduction to the Theory of Numbers*. Wiley, New York, 1972.
8. Pohlig, S.C., and Hellman, M.E. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. To appear in IEEE Trans. Inform. Theory, 1978.
9. Pollard, J.M. Theorems on factorization and primality testing. *Proc. Camb. Phil. Soc. 76* (1974), 521–528.
10. Potter, R.J., Electronic mail. *Science 195,* 4283 (March 1977), 1160–1164.
11. Rabin, M.O., Probabilistic algorithms. In *Algorithms and Complexity*, J. F. Traub, Ed., Academic Press, New York, 1976, pp. 21–40.
12. Solovay, R., and Strassen, V. A Fast Monte-Carlo test for primality. *SIAM J. Comptng. 6* (March 1977), 84–85.
13. *Federal Register, Vol. 40*, No. 52, March 17, 1975.
14. *Federal Register, Vol. 40*, No. 149, August 1, 1975.

(A comment on this article may be found in the Technical Correspondence section of this issue, page 173.—Ed.)