MIT 6.875

Foundations of Cryptography
Lecture 7

The Goldreich-Levin (GL) Theorem

Let $\{B_r: \{0,1\}^n \to \{0,1\}\}$ where

$$B_r(x) = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i \bmod 2$$

be a collection of predicates (one for each r). Then, a ***random*** B_r is hardcore for ***every*** one-way function F. That is, for every one-way function F, every PPT A, there is a negligible function μ s.t.

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n: A(F(x), r) = B_r(x)] \leq \frac{1}{2} + \mu(n)$$

GL Theorem: Alternative Interpretation

For ***every*** one-way function/permutation F, there is a related one-way function/permutation

$$F'(x, r) = (F(x), r)$$

which has a *deterministic* hardcore predicate. In particular, the predicate $B(x, r) = \langle r, x \rangle \bmod 2$ is hardcore for F'.

$$\Pr\left[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : A\left(F'(x, r)\right) = \langle r, x \rangle\right] \leq \frac{1}{2} + \mu(n)$$

<u>Key Point</u>:

This statement is *sufficient* to construct PRGs from any OWP.

A Proof of the GL Theorem

Assume (after averaging) that for $\geq 1/2p(n)$ fraction of the x,

$$\Pr[r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x \rangle] \geq \mathbf{\frac{1}{2}} + 1/2p(n)$$

Recall the template:

Pick a random r, ask P for $\langle r + e_i, x \rangle$ ~~and $\langle r, x \rangle$.~~ Subtract the two to get $\langle e_i, x \rangle = x_i$. **and guess $\langle r, x \rangle$**

The idea: Parsimony in Guessing

Pick random "seed vectors" $s_1, \dots, s_{\log(m+1)}$, and **guess** $c_j = \langle s_j, x \rangle$ for all j.

The probability that all guesses are correct is $\frac{1}{2^{\log(m+1)}} = 1/(m+1)$ which is not bad.

From the seed vectors, generate many more r_i.

Let $T_1, \dots, T_m$ denote all possible non-empty subsets of $\{1,2,\dots,\log(m+1)\}$. We will let

$$r_i = \bigoplus_{j \in T_i} s_j \qquad \text{and} \qquad b_i = \bigoplus_{j \in T_i} c_j$$

Key Observation: If the guesses $c_1, \dots, c_{\log(m+1)}$ are all correct, then so are the $b_1, \dots, b_m$.

The OWF Inverter

Generate random $s_1, \ldots, s_{\log(m+1)}$ and bits $c_1, \ldots, c_{\log(m+1)}$.

From them, derive $r_1, \ldots, r_{\log(m+1)}$ and bits $b_1, \ldots, b_m$ as in the previous slide.

Repeat for each $i \in \{1,2, \ldots, n\}$:

Repeat $\mathbf{O}\left(\boldsymbol{n}\left(\boldsymbol{p}(\boldsymbol{n})\right)^{\mathbf{2}}\right)$ times:

Ask P to tells us $\langle r_{ij} + e_i, x\rangle$. XOR P's reply with b_i to get a guess for x_i.

Compute the majority of all such guesses and set the bit as x_i

Output the concatenation of all x_i as x.

Analysis of the Inverter

Let's condition on the guesses $c_1, \ldots, c_{\log(m+1)}$ being all correct.

The main issue: The r_i are not independent (can't do Chernoff)

Key Observation: The r_i **are** pairwise independent.

Therefore, can apply Chebyshev!

We will show that
$p := \Pr[\text{Inverter succeeds} \mid \text{all guesses correct}, \text{good x}] \geq 0.98.$

(We will prove this in a bit, but let's assume it for now)

Finishing the proof (assuming p is large)

Pr[Inverter succeeds]
≥ Pr[Inverter succeeds | all guesses correct, good x] ·
Pr[all guesses correct] · Pr[good x]

$$= p \cdot \frac{1}{m+1} \cdot \frac{1}{2p(n)} = p \cdot \frac{1}{2n^2 p(n)^3}$$

So, it suffices to show that p is large.

We will now show that $p \geq 0.98$, so we are done.

Can also make the success probability $\approx 1/p(n)$ by enumerating over all the "guesses". Each guess results in a supposed inverse, but we can check which of them is the actual inverse!

Analysis of the Inverter: Estimating p

The probability that a single iteration of the inner loop gives the correct x_i is at least $\frac{1}{2} + 1/2p(n)$.

Let this be the good event E_i (for the i^{th} iteration of the inner loop).

The majority decision is correct if the number of events E_i that occur is at least $\frac{m}{2} = 50\, n(p(n))^2$.

The expected number of events that occur is

$$(\frac{1}{2} + \frac{1}{2p(n)}).\ 100\, n(p(n))^2 = 50\, n(p(n))^2 + 50np(n).$$

The variance is

$$\approx \frac{1}{2} \cdot 100\, n\big(p(n)\big)^2 = 50n\big(p(n)\big)^2$$

Analysis of the Inverter: Estimating p

The expected number of events that occur is

$$(\frac{1}{2} + \frac{1}{2p(n)}).\ 100\, n(p(n))^2 = 50\, n(p(n))^2 + 50np(n).$$

The variance is $\approx \frac{1}{2} \cdot 100\, n\big(p(n)\big)^2 = 50n\big(p(n)\big)^2$

By an application of Chebyshev, we have

$$\Pr[majority\ decision\ w.r.t\ x_i\ incorrect] \leq \frac{50n\big(p(n)\big)^2}{\big(50np(n)\big)^2} = \frac{1}{50n}$$

By an application of union bound, we have

$$\Pr[one\ of\ the\ x_i\ is\ incorrect] \leq n \cdot \frac{1}{50n} = 1/50$$

∴ The inverter outputs the correct inverse w.p. $\mathrm{p} \geq 0.98$.

The Coding-Theoretic View of GL

$x \to (\langle x, r \rangle)_{r \in \{0,1\}^n}$ can be viewed as a highly redundant, exponentially long encoding of x = **the Hadamard code**.

$P(F(x), r)$ can be thought of as providing access to a **noisy** codeword.

What we proved = **unique decoding** algorithm for Hadamard code with error rate $\frac{1}{4} - 1/p(n)$.

The real proof = **list-decoding algorithm** for Hadamard code with error rate $\frac{1}{2} - 1/p(n)$.

GL as Randomness Extraction

Randomness extractor: You get an n-bit string from a "source" (probability distribution) with large (min-)entropy, say $n/2$. Can you run a *deterministic* procedure that outputs a single, nearly uniformly random, bit?

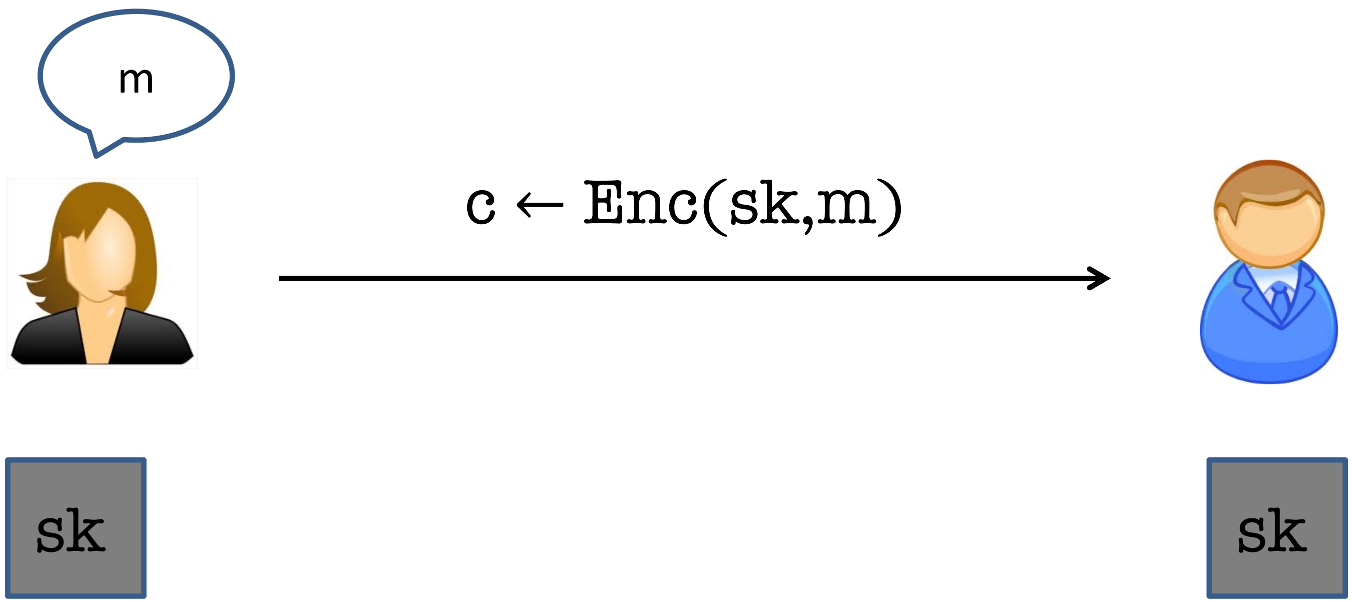
This is impossible.

But possible given a short, truly random, "seed". Leads to an entire theory of randomness extraction.

Back to our setting… the distribution of X conditioned on $f(X)$ does not have any entropy (since f is a permutation) but it does have "computational entropy". Can we extract a computationally random bit from it? This is what GL does!

Secret-Key Encryption

(also called symmetric encryption)



The Key Agreement Problem:

How did Alice and Bob get the same sk to begin with?!

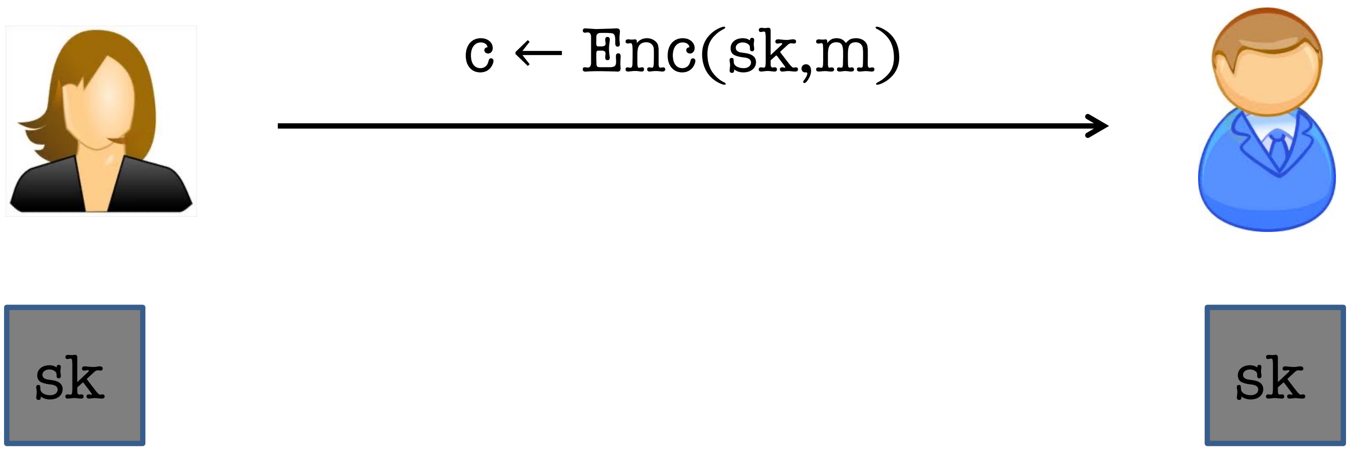
Secret-Key Encryption

The Key Agreement Problem:
How did Alice and Bob get the same $\mathtt{sk}$ to begin with?!

Physical Exchange of Keys is Clunky and Impractical:

- What if Alice and Bob have never met in person?
- Even so, what if they need to refresh their keys?
- Too expensive and cumbersome:
  Each user will need to store N keys, too expensive!

Secret-Key Encryption



The Key Agreement Problem: Can Alice and Bob, who *never previously met*, exchange messages securely?

The Next Few Lectures

- Key Agreement and Public-key Encryption:
  Definition and Properties

- Constructions

  1: Diffie-Hellman/El Gamal

  2: Trapdoor Permutations (RSA)

  3: Quadratic Residuosity/Goldwasser-Micali

  4: Learning with Errors/Regev

Merkle (1974)

C.S. 244
FALL 1974

Project 2 looks more reasonable, maybe because your description Project 1 is muddled terribly. Talk to me about these today.

Ralph Merkle

Project Proposal

Topic: Establishing secure communications between seperate secure sites over insecure communication lines.

Assumptions: No prior arrangements have been made between the two sites, and it is assumed that any information known at either site is known to the enemy. The sites, however, are now secure, and any new information will not be divulged.

Method 1: Guessing. Both sites guess at keywords. These guesses are one-way encrypted, and transmitted to the other site. If both sites should chance to guess at the same keyword, this fact will be discovered when the encrypted versions are compared, and this keyword will then be used to establish a communications link.

Discussion: No, I am not joking. If the keyword space is of size N, then the probability that both sites will guess at a common keyword rapidly approaches one after the number of guesses exceeds sqrt(N). Anyone listening in on the line must examine all N possibilities. In more concrete

I believe that it is possible for two people to communicate securely without having made any prior arrangements that are not completely public. My quarter project would be to investigate any method by which this could be accomplished, and what advantages and disadvantages these methods might have over other ways of establishing secure communications.

Programming Techniques

S. L. Graham, R. L. Rivest
Editors

Secure Communications Over Insecure Channels

Ralph C. Merkle
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

According to traditional conceptions of cryptographic security, it is necessary to transmit a key, by secret means, before encrypted messages can be sent securely. This paper shows that it is possible to select a key over open communications channels in such a fashion that communications security can be maintained. A method is described which forces any enemy to expend an amount of work which increases as the square of the work required of the two communicants to select the key. The method provides a logically new kind of protection against the passive eavesdropper. It suggests that further research on this topic will be highly rewarding, both in a theoretical and a practical sense.

Key Words and Phrases: security, cryptography, cryptology, communications security, wiretap, computer network security, passive eavesdropping, key distribution, public key cryptosystem

CR Categories: 3.56, 3.81

Merkle's Idea

$$\{H(x_1), H(x_2), \ldots, H(x_n)\}$$
$$\longrightarrow$$

$$\{H(y_1), H(y_2), \ldots, H(y_n)\}$$
$$\longleftarrow$$

Pick n random numbers $x_1, \ldots, x_n$

Pick n random numbers $y_1, \ldots, y_n$

Assume that $H: [n^2] \to [n^2]$ is an injective OWF.

Assume that $H: [n^2] \to [n^2]$ is an injective OWF.

Merkle's Idea

$\{\boldsymbol{H(x_1)}, H(x_2), \ldots, H(x_n)\}$ $\longrightarrow$

$\longleftarrow$ $\{H(y_1), \boldsymbol{H(y_2)}, \ldots, H(y_n)\}$

Pick n random numbers $x_1, \ldots, x_n$

Pick n random numbers $y_1, \ldots, y_n$

There is a common number (say $x_i = y_j$ w.h.p.)

Alice and Bob can detect it in time $O(n)$, and they set it as their shared key.

Assume that $H: [n^2] \to [n^2]$ is an injective OWF.

Merkle's Idea

$$\{\boldsymbol{H(x_1)}, H(x_2), \dots, H(x_n)\} \longrightarrow$$

$$\longleftarrow \{H(y_1), \boldsymbol{H(y_2)}, \dots, H(y_n)\}$$

Pick n random numbers $x_1, \dots, x_n$

Pick n random numbers $y_1, \dots, y_n$

How long does it take **Eve** to compute the shared key?

She knows i and j, but she needs to invert the OWF. Assuming the OWF is very strong, that is $\Omega(n^2)$ time!

Assume that $H: [n^2] \to [n^2]$ is an injective OWF.

Merkle's Idea

$$\{\boldsymbol{H(x_1)}, H(x_2), \ldots, H(x_n)\} \longrightarrow$$

$$\longleftarrow \{H(y_1), \boldsymbol{H(y_2)}, \ldots, H(y_n)\}$$

Pick n random numbers $x_1, \ldots, x_n$

Pick n random numbers $y_1, \ldots, y_n$

Problem: Only protects against quadratic-time Eves (still an excellent idea)

644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.

Manuscript received June 3, 1976. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173. Portions of this work were presented at the IEEE Information Theory Workshop, Lenox, MA, June 23–25, 1975 and the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21–24, 1976.

W. Diffie is with the Department of Electrical Engineering, Stanford University, Stanford, CA, and the Stanford Artificial Intelligence Laboratory, Stanford, CA 94305.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a multiple access cipher. A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other enciphered in the receiver's public enciphering key and deciphers the messages he receives using his own secret deciphering key.

We propose some techniques for developing public key cryptosystems, but the problem is still largely open.

Public key distribution systems offer a different approach to eliminating the need for a secure key distribution channel. In such a system, two users who wish to exchange a key communicate back and forth until they arrive at a key in common. A third party eavesdropping on this exchange must find it computationally infeasible to compute the key from the information overheard. A possible solution to the public key distribution problem is given in Section III, and Merkle [1] has a partial solution of a different form.

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary busi-

Diffie & Hellman 1976

Marked the birth of public-key cryptography.

Invented the Diffie-Hellman key exchange (conjectured to be secure against all poly-time attackers *unlike* Merkle).

Used to this day (e.g., TLS 1.3) albeit with different groups than what DH had in mind.

Turing Award 2015

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 $e * d \equiv 1 (\bmod (p - 1) * (q - 1))$. 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

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.

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

$$E(D(M)) = M. \tag{2}$$

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

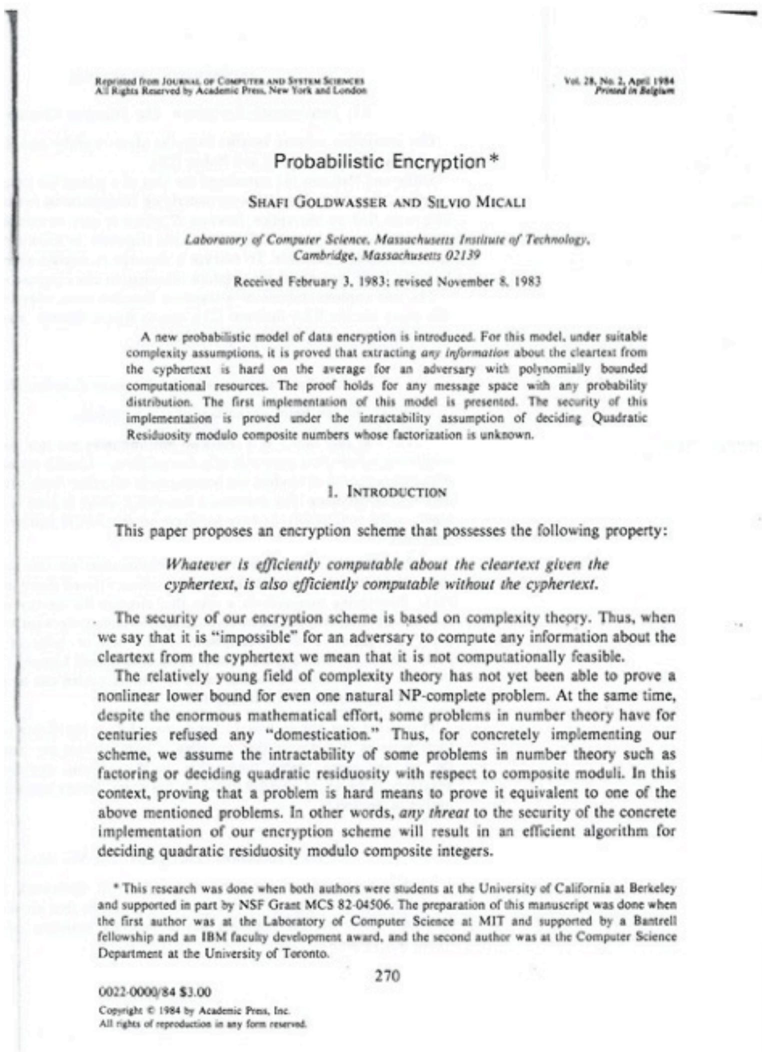
Rivest, Shamir & Adleman 1978

Invented the RSA trapdoor permutation, public-key encryption and digital signatures.

RSA Signatures used to this day (e.g., TLS 1.3) in essentially the original form it was invented.

Turing Award 2002

Goldwasser & Micali 1982



Reprinted from JOURNAL OF COMPUTER AND SYSTEM SCIENCES
All Rights Reserved by Academic Press, New York and London

Vol. 28, No. 2, April 1984
Printed in Belgium

Probabilistic Encryption*

SHAFI GOLDWASSER AND SILVIO MICALI

Laboratory of Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Received February 3, 1983; revised November 8, 1983

A new probabilistic model of data encryption is introduced. For this model, under suitable complexity assumptions, it is proved that extracting *any information* about the cleartext from the cyphertext is hard on the average for an adversary with polynomially bounded computational resources. The proof holds for any message space with any probability distribution. The first implementation of this model is presented. The security of this implementation is proved under the intractability assumption of deciding Quadratic Residuosity modulo composite numbers whose factorization is unknown.

1. INTRODUCTION

This paper proposes an encryption scheme that possesses the following property:

Whatever is efficiently computable about the cleartext given the cyphertext, is also efficiently computable without the cyphertext.

The security of our encryption scheme is based on complexity theory. Thus, when we say that it is "impossible" for an adversary to compute any information about the cleartext from the cyphertext we mean that it is not computationally feasible.

The relatively young field of complexity theory has not yet been able to prove a nonlinear lower bound for even one natural NP-complete problem. At the same time, despite the enormous mathematical effort, some problems in number theory have for centuries refused any "domestication." Thus, for concretely implementing our scheme, we assume the intractability of some problems in number theory such as factoring or deciding quadratic residuosity with respect to composite moduli. In this context, proving that a problem is hard means to prove it equivalent to one of the above mentioned problems. In other words, *any threat* to the security of the concrete implementation of our encryption scheme will result in an efficient algorithm for deciding quadratic residuosity modulo composite integers.

* This research was done when both authors were students at the University of California at Berkeley and supported in part by NSF Grant MCS 82-04506. The preparation of this manuscript was done when the first author was at the Laboratory of Computer Science at MIT and supported by a Bantrell fellowship and an IBM faculty development award, and the second author was at the Computer Science Department at the University of Toronto.

0022-0000/84 $3.00

"Probabilistic Encryption": defined what is now the gold-standard of security for public-key encryption (two *equivalent* defs: indistinguishability and semantic security)

GM-encryption: based on the difficulty of the quadratic residuosity problem, the first homomorphic encryption.

Turing Award 2012

The Secret History of Public-key Encryption

Claimed to be invented in secret in early 1970s at the GCHQ (British NSA) by James Ellis, Clifford Cocks and Malcolm Williamson.

THE STORY OF NON-SECRET ENCRYPTION

by J H ELLIS

1. Public-key cryptography (PKC) has been the subject of much discussion in the open literature since Diffie and Hellman suggested the possibility in their paper of April 1976 (reference 1) . It has captured public imagination, and has been analysed and developed for practical use. Over the past decade there has been considerable academic activity in this field with many different schemes being proposed and, sometimes, analysed.

2. Cryptography is a most unusual science. Most professional scientists aim to be the first to publish their work, because it is through dissemination that the work realises its value. In contrast, the fullest value of cryptography is realised by minimising the information available to potential adversaries. Thus professional cryptographers normally work in closed communities to provide sufficient professional interaction to ensure quality while maintaining secrecy from outsiders. Revelation of these secrets is normally only sanctioned in the interests of historical accuracy after it has been demonstrated clearly that no further benefit can be obtained from continued secrecy.

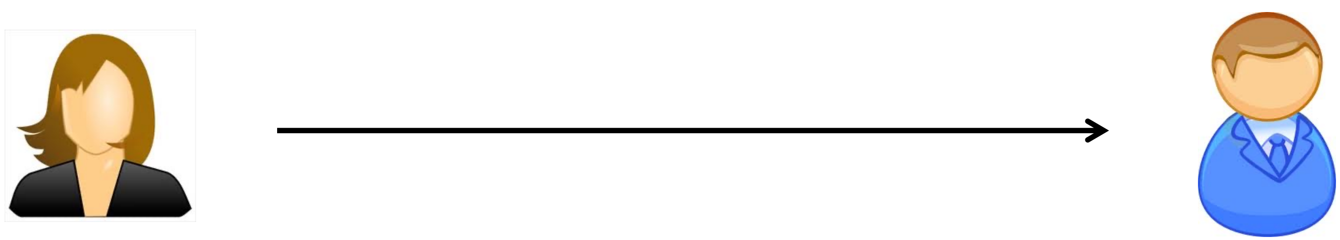
3. In keeping with this tradition it is now appropriate to tell the story of the invention and development within CESG of non-secret encryption (NSE) which was our original name for what is now called PKC. The task of writing this paper has devolved on me because NSE was my idea and I can therefore describe these early developments from personal experience. No techniques not already public knowledge, or specific applications of NSE will be mentioned. Neither shall I venture into evaluation. This is a simple, personal account of the salient features, with only the absolute minimum of mathematics.

4. The story begins in the 60's. The management of vast quantities of key material needed for secure communication was a headache for the armed forces. It was obvious to everyone, including me, that no secure communication was possible without secret key, some other secret knowledge, or at least some way in which the recipient was in a different position from an interceptor. After all, if they were in identical situations how could one possibly be able to receive what the other could not? Thus there was no incentive to look for something so clearly impossible.

5. The event which changed this view was the discovery of a wartime, Bell-Telephone report by an unknown author describing an ingenious idea for secure telephone speech (reference 2). It proposed that the recipient should mask the sender's speech by adding noise to the line. He could subtract the noise afterwards since he had added it and therefore knew what it was. The obvious practical disadvantages of this system prevented it being actually used, but it has some interesting characteristics. One of these, irrelevant to the main theme, is the

Public-Key Encryption

(also called ***a***symmetric encryption)



GOAL:

Anyone **can encrypt to Bob.**

Bob, and only Bob, can decrypt.

Bob	pk

Public-Key Encryption

$m \leftarrow \mathrm{Dec}(\mathbf{sk}, c)$

$c \leftarrow \mathrm{Enc}(\mathbf{pk}, m)$

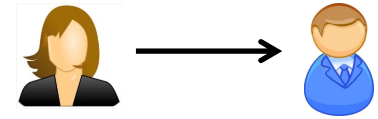
sk

❶ Bob generates a pair of keys, a public key **pk**, and a private (or secret) key **sk**.

❷ Bob “publishes” **pk** and keeps **sk** to himself.

❸ Alice encrypts m to Bob using **pk**

❹ Bob decrypts using **sk**

Public-Key Encryption 

A triple of PPT algorithms (Gen, Enc, Dec) s.t.

- $(pk, sk) \leftarrow Gen(1^n)$. 

PPT Key generation algorithm generates a public-private key pair.

- $c \leftarrow Enc(pk, m)$. 

Encryption algorithm uses the public key to encrypt message m.

- $m \leftarrow Dec(sk, c)$. 

Decryption algorithm uses the private key to decrypt ciphertext c.

Correctness: For all pk, sk, m: $Dec\big(sk, Enc(pk, m)\big) = \mathrm{m}$.

How to Define Security

Bob	pk

$c \leftarrow \mathrm{Enc}(\mathbf{pk}, m)$

sk

Eve knows Bob's public key pk

Eve sees polynomially many ciphertexts $c_1, c_2, \ldots$ of messages $m_1, m_2, \ldots$

Given this: Eve should not get *any partial information* about the set of messages.

IND-Security (also called IND-CPA)

Challenger

Eve

$(pk, sk) \leftarrow Gen(1^n)$ $\xrightarrow{\quad pk \quad}$

$\overrightarrow{m_0} = (m_0^1, m_0^2, \ldots, m_0^L)$

$\longleftarrow$

$\overrightarrow{m_1} = (m_1^1, m_1^2, \ldots, m_1^L)$

$\boldsymbol{s.t.\ |m_0^i| = |m_1^i|\ for\ all\ i}$

$b \leftarrow \{0,1\}$

$c_i \leftarrow Enc(pk, m_b^i)$ $\xrightarrow{\quad (c_1, c_2, \ldots, c_L) \quad}$

$\xleftarrow{\quad b' \quad}$

Eve wins if $b' = b$. The encryption scheme is IND-secure if no PPT Eve can win in this game with probability better than $\frac{1}{2} + \mathrm{negl}(n)$.

An Alternative Definition

"Semantic Security": the computational analog of Shannon's perfect secrecy definition.

Turns out to be equivalent to IND-security (just as in Lec 1 but the proof is more complex)

We will stick to IND-security as it's easy to work with.

Simplifying the Definition: One Message to Many Message Security

Challenger

Eve

Challenger		Eve
$(pk, sk) \leftarrow Gen(1^n)$	$\xrightarrow{\quad pk \quad}$	
$b \leftarrow \{0,1\}$	$\xleftarrow{\quad \boldsymbol{m_0, m_1 \ \ s.t.\ |m_0| = |m_1|} \quad}$	
$\boldsymbol{c \leftarrow Enc(pk, m_b; r)}$	$\xrightarrow{\quad \boldsymbol{c} \quad}$	
	$\xleftarrow{\quad b' \quad}$	

Eve wins if $b' = b$. The encryption scheme is single-message-IND-secure if no PPT Eve can win with prob. better than $\frac{1}{2} + \text{negl}(n)$.

Simplifying the Definition: One Message to Many Message Security

Challenger

Eve

$(pk, sk) \leftarrow Gen(1^n)$

pk →

$m_0, m_1 \quad s.t.\ |m_0| = |m_1|$ ←

$b \leftarrow \{0,1\}$

$c \leftarrow Enc(pk, m_b)$

$\boldsymbol{c}$ →

b' ←

Theorem: A public-key encryption scheme is IND-secure iff it is single-message IND-secure.

Constructions of Public-key Encryption

1: Diffie-Hellman/El Gamal

2: Trapdoor Permutations (RSA)

3: Quadratic Residuosity/Goldwasser-Micali

4: Learning with Errors/Regev

Groups

Group G: (finite set S, group operation $*: S \times S \to S$)

Associative: $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$

Commutative: $g_1 * g_2 = g_2 * g_1$

Identity: $\mathrm{Id} * g = g * Id = g$

Inverse: for every g, there is a g' s.t.

$$g * g' = g' * g = Id$$

Order

Order of a Group G = $(S, *)$ is simply $|S|$
(sometimes we will just write $|G|$).

Order of an element g ∈ G , denoted $ord(g)$ is the minimum $n > 0$ s.t.

$$g * g * \cdots * g = Id$$

Lagrange's Theorem: $ord(g)$ always divides $|G|$.

A **generator** is an element of order $|G|$.

A **cyclic group** is one that has a generator.

The Additive Group $\mathbb{Z}_N$

$\mathbb{Z}_N$: $(\{0,1,\ldots,\mathrm{N}-1\}$, group operation: + mod N)

- Order?
- Generators?

The Additive Group $\mathbb{Z}_N$

$\mathbb{Z}_N$: $(\{0,1,\ldots,\mathrm{N}-1\}$, group operation: + mod N)

- Computing the group operation is easy
  ($=\mathrm{poly}(\log N)$ time).

- Computing inverses is easy.

- Iterated group operations ("exponentiation")

  Given $g \in \mathbb{Z}_N$ and $n \in \mathbb{Z}$, compute $\underbrace{g+g+\cdots+g}_{n \text{ times}}$

The Additive Group $\mathbb{Z}_N$

$\mathbb{Z}_N$: $(\{0,1,\ldots,\mathrm{N}-1\}$, group operation: + mod N)

- Computing the group operation is easy (=poly time).
- Computing inverses is easy.
- Exponentiation is easy.
- The discrete logarithm problem is

Given $g, h \in \mathbb{Z}_p$, find $n \in \mathbb{Z}$, s.t. $h = \underbrace{g + g + \cdots + g}_{n \text{ times}}$

$= ng \pmod N$

Extended Euclidean algorithm: $n = hg^{-1} (\mathrm{mod}\ N)$

The Multiplicative Group $\mathbb{Z}_p^*$

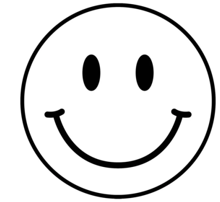
$\mathbb{Z}_p^*$: $(\{1, \ldots, \mathrm{p}-1\}$, group operation: $\cdot$ mod p): **p prime**

- Order the group $= \varphi(p) = p - 1$

  (Euler's totient function $\varphi(N) = |\{1 \leq \mathrm{x} < \mathrm{N} : \mathrm{gcd}(\mathrm{x}, \mathrm{N}) = 1\}|$

- If p is prime, $\mathbb{Z}_p^*$ is cyclic.

The Multiplicative Group $\mathbb{Z}_p^*$



$\mathbb{Z}_p^*$: $(\{1, \ldots, \mathrm{p} - 1\}$, group operation: $\cdot$ mod p)

- Computing the group operation is easy.
- Computing inverses is
- Exponentiation (given $g \in \mathbb{Z}_p^*$ and $x \in \mathbb{Z}_{p-1}$, find g^x mod p)
- The discrete logarithm problem given $g, h \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_{p-1}$ s.t. $\mathrm{h} = g^x$ mod p) is

Diffie-Hellman Key Exchange

Commutativity in the exponent: $(g^x)^y = (g^y)^x$

(where g is an element of some group)

So, you can compute g^{xy} given either g^x and y, or g^y and x.

Diffie-Hellman Assumption (DHA):

Hard to compute g^{xy} given only g, g^x and g^y

Diffie-Hellman Key Exchange

Diffie-Hellman Assumption (DHA):

Hard to compute it given only g, g^x and g^y

We know that if discrete log is easy, DHA is false.

Major Open Problem:

Are discrete log and DHA equivalent?

Diffie-Hellman Key Exchange

p, g: Generator of our group Z_p^*

$g^x \bmod p$

$g^y \bmod p$

Pick a random number $x \in Z_{p-1}$

Pick a random number y $\in Z_{p-1}$

Shared key K = $g^{xy} \bmod p$

$= (g^y)^x \bmod p$

Shared key K = $g^{xy} \bmod p$

$= (g^x)^y \bmod p$

Diffie-Hellman/El Gamal Encryption

- $Gen(1^n)$: Generate an n-bit prime p and a generator g of Z_p^*. Choose a random number $x \in Z_{p-1}$

  Let $pk = (p, g, g^x)$ and let $sk = x$.

- $Enc(pk, m)$ where $m \in Z_p^*$: Generate random $y \in Z_{p-1}$ and output $(g^y, g^{xy} \cdot m)$

- $Dec(sk = x, c)$: Compute g^{xy} using g^y and x and divide the second component to retrieve m.

How to make this really work? **Is this Secure?**

How to come up with a prime p

- How to come up with a group $\mathbb{Z}_p^*$ = how to generate a large prime p?

  (1) **Prime number theorem**: $\approx 1/n$ fraction of $\boldsymbol{n}$**-bit** numbers are prime.

  (2) **Primality tests** [Miller'76, Rabin'80, Agrawal-Kayal-Saxena'02] Can test in time poly(n) if a given n-bit number is prime.

How to come up with a generator g

- If p is prime, $\mathbb{Z}_p^*$ is cyclic (so generators exist).
- How to come up with a generator of $\mathbb{Z}_p^*$?

(1) **There are lots of generators**: $\approx 1/n$ fraction of $\boldsymbol{n}$-**bit** numbers are prime.

(2) **Testing if $\boldsymbol{g}$ is a generator**:

Theorem: let $q_1, \ldots, q_k$ be the prime factors of p. Then, g is a generator of $\mathbb{Z}_p^*$ if and only if

$g^{(p-1)/q_i} \neq 1 \pmod{p}$ for all i.

To Summarize

- Pick a random prime p *together with* its prime factorization (Adam Kalai 2000 paper)
- Pick a random element of $\mathbb{Z}_p^*$ and test if it is a generator (using theorem from last slide).
- Continue this process until you hit a generator. The density of generators is large enough that this will converge in expected poly(log p) time.
- Another, more commonly used method, in the next lecture.

Next Lecture: More on Diffie-Hellman Key Exchange