Heraclitean Encryption

Michael Ernst Gideon Yuval

October 27, 1993
Revised June 29, 1994

Technical Report
MSR-TR-94-13

Microsoft Research
Advanced Technology Division
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Authors' email: mernst@research.microsoft.com, gideony@microsoft.com.

Abstract

Most encryption schemes always use the same decryption key to convert a particular codetext into plaintext. If a decryption key that has been revealed to multiple parties is compromised, it is impossible to determine who is responsible for the breach. Heraclitean encryption, which uses public-key encryption (for instance, RSA or elliptic curve) as its cryptographic basis, permits the encryptor to create as many independent decryption keys as desired. Each decryption key can publicly encode information about the party to whom it was issued, so that given a key, anyone can determine its owner. Since decryption keys can be traced, their holders have an incentive to keep them secret.

We discuss applications of Heraclitean encryption, provide an example implementation, discuss weaknesses in that implementation, and explore some practicalities of using the scheme.

We do not address the issue of tracking decrypted information back to the decryptor; the plaintext is identical for each recipient. Heraclitean encryption is applicable to any broadcast medium that can carry proprietary information—for instance, pay-per-view video and wide distribution of commercial software or databases via CD-ROM or bulletin boards.

1 Introduction

Consider a company that widely distributes its software in encrypted form: a single readily-available CD-ROM holds dozens or hundreds of applications. To purchase a software package, a customer provides a name and credit card number to the software company, receives a decryption key in return, and passes the key to an installation program. The key can then be discarded and the unencrypted software used in the normal way. This scenario is appealing because buying software becomes extremely convenient, accruing advantages to both the seller and the purchaser.

The downside of this scenario is that software theft potentially becomes even easier than it is today. Current public-key and private-key encryption schemes provide for a single decryption key which is provided to every bona fide decryptor. If a software package's decryption key is anonymously revealed by some unscrupulous purchaser, the software company has no way of determining which of its customers is at fault. Assuming that all pirates have copies of the encrypted CD-ROM, they need only communicate (for instance, place on a bulletin board system) the decryption key instead of the entire program. The software company would like to either make this impossible or provide a disincentive to those who would reveal keys. One way to do so is to make every decryption key unique and traceable to its rightful holder.

Making decryption keys unique is easy if the codetext can be tailored to that key, but that is often impractical. For instance, CD-ROMs are produced by a printing process, so every one is identical. While they can be burned with a laser for identification purposes, it is not hard to fake another burn pattern. Other broadcast media, such as cable and pay-per-view television, also preclude customization of the signal for each recipient.

Heraclitean encryption permits distinct decryption keys to decode a single codetext. Furthermore, additional information (such as the purchaser's name and credit card number) can be encoded in the decryption key in such a way that revealing the decryption key reveals the additional information. This creates two incentives for a purchaser not to distribute the key, enabling others to obtain the encrypted information. First, since a key indicates the party to whom it was assigned, the revealer is susceptible to legal proceedings for piracy. Second, if the purchaser's credit card number is encoded in the decryption key, revealing the key makes the revealer vulnerable to credit card fraud.

The software-on-CD-ROM scenario is just one application of Heraclitean encryption. It can be used whenever the rightful recipients of broadcasted encrypted information have less incentive to keep it secret than the sender but do not want to be seen to be publicly distributing the information. Heraclitean encryption can also be used to trace unintentional leaks.

The Greek philosopher Heraclitus [Her] noted that, because a river constantly changes, no one can step into the same river twice:

> You could not step twice into the same rivers;
> For other waters are ever flowing onto you.

We call our scheme Heraclitean because its goal is to prevent any two distinct decryption keys from sharing any information.

2 Tying keys to users

For concreteness, our examples use the RSA cryptosystem [RSA78]. Let $n = pq$, where p and q are prime, and choose e and d such that $ed \bmod \phi(n) = 1$, where ϕ is Euler's totient function. In this case, $\phi(n) = (p-1)(q-1)$. The encryption function is $E(a) = a^e \bmod n$ and the decryption function is $D(a) = a^d \bmod n$. The modulus n is made public, but neither e nor d is revealed.

Let *id* be some identifying information, and let x be a numeric representation of *id* from which *id* can be uniquely recovered. The encryptor computes, modulo $\phi(n)$, user key $y = d/x$, so that $xy \bmod \phi(n) = d$. User key y is useless without x, but together they can be used for decryption:

$$exy = 1 \pmod{\phi(n)} ,$$

so

$$((E(a))^x)^y = a^{exy} = a \pmod{n} .$$

(This is not the final scheme; it contains a hole that the interested reader will find before proceeding to the next section.)

It is easy to identify anyone who reveals x and y, thereby letting others decipher the codetext, because x uniquely specifies *id*. Revealing xy also permits a to be determined from $E(a)$, so x should be uniquely determinable from xy, not just for the issuer of x and y, but for anyone.

We accomplish this by encoding the identifying information id in x using a Gödel encoding [LP81] followed by an error-correcting coding (ECC). Our Gödel numbering associates with each string a number in $\mathbf{Z}_{\phi(n)}$ with no large factors. For instance, the lexically 1984th string might be represented by the Gödel number $2^4 \cdot 3^8 \cdot 5^9 \cdot 7^1$. (Actually, that particular encoding is unacceptable because $2 \mid \phi(n)$; the primes $p \not\mid \phi(n)$ would be used as the exponentiation bases.)

Such an encoding is reversible, because the resulting number is easy to factor. If y has no small factors, then x is the product of the small factors of xy. In general, y may have small factors, so error-correcting coding is used in order to make x uniquely determinable from xy. The ECC problem is especially tractable in this case, because errors occur in one direction only (multiplication by y can add small factors, but cannot remove any) and because the erroneous values for each factor are likely to differ only slightly from the intended values (y is likely to have only a few small factors).

The encryptor can slightly modify x (say, by adding a noise character to the end of id) if x is not uniquely determinable from xy (that is, if the error-correcting coding fails). Following this rule guarantees that no two user keys y, y' are issued such that $xy = x'y'$.

The exact encoding used, including techniques used to reduce its size (such as compression of the identifying string id), is an implementation detail.

3 Removing the relationship between keys

The scheme described above can be broken, because different decryption keys are simply different factorings of the mod-$\phi(n)$ inverse of e. This section describes the attack and presents a fix for it.

Two purchasers with identification numbers x_1, x_2 and user keys y_1, y_2 have

$$x_1 y_1 = x_2 y_2 \pmod{\phi(n)} ,$$

which implies

$$\phi(n) \mid x_1 y_1 - x_2 y_2 .$$

Most arbitrarily generated numbers can be easily factored. The factorization of $x_1 y_1 - x_2 y_2$ yields $\phi(n)$, from which $d = x_1 y_1 \bmod \phi(n) = x_2 y_2 \bmod \phi(n)$ can be computed and distributed with impunity. (Given $\phi(n)$, it is also

possible to compute e and n.) If three purchasers collude, they have

$$\phi(n) \mid \gcd(x_1y_1 - x_2y_2, x_1y_1 - x_3y_3, x_2y_2 - x_3y_3) \ ,$$

and computation of $\phi(n)$ is even easier.

We introduce randomness in order to prevent different keys from relating to one another in any direct way.

Two encryptions, $E(a) = a^e \bmod n$ and $E'(a) = a^{e'} \bmod n$, are publicized. Given Gödel encodings x and x' of the identifying information, select a random value $r \in \mathbf{Z}_{\phi(n)}$ and choose decryption keys y and y' to satisfy

$$\begin{aligned} exy &= r \pmod{\phi(n)} \\ e'x'y' &= 1 - r \pmod{\phi(n)} \ , \end{aligned}$$

which implies

$$exy + e'x'y' = 1 \pmod{\phi(n)} \ .$$

Codetext pair $\langle E(a) = a^e \bmod n, E'(a) = a^{e'} \bmod n \rangle$ is decrypted by computing

$$((a^e)^x)^y \cdot ((a^{e'})^{x'})^{y'} = a^{exy + e'x'y'} = a \pmod{n} \ .$$

A new random value r is generated for each pair $\langle y, y' \rangle$ of decryption keys. It is possible, but not necessary, to choose r so as to give y or y' (or both) special properties, such as absence of small factors, to simplify recovery of x and x' from xy and $x'y'$.

4 Worked examples

This section presents examples of the schemes presented above, in order to make them more concrete.

Throughout, let the secret primes be $p = 11$, $q = 17$; their product, $n = 187$, is publicized. (In practice, very much larger primes would be used, so that factoring the product is computationally infeasible.) The euler totient $\phi(n) = 10 * 16 = 160$ is secret and will be used by the encryptor in computing other values.

Let the plaintext (the information to be distributed in encrypted form) be $a = 10$, and arbitrarily choose encryption exponent $e = 123$. The encrypted information (available to all potential customers) is $E(a) = a^e \bmod n = 10^{123} \bmod 187 = 54$. The encryptor computes universal decryption exponent $d = e^{-1} = 147 \pmod{\phi(n)}$, but keeps it secret.

4.1 Single encryption scheme

Let the unique user identifier be computed by using using a Gödel encoding which exponentiates successive primes according to the (ordinal positions of) characters in the user's name. (The primes 2 and 5 are omitted because they divide $\phi(n)$.) We omit the error-correcting coding for this example, to keep it simple.

Suppose customer "BC" buys code from the encryptor. BC's Gödel number $x = 3^2 \cdot 7^3 = 3087$. The encryptor computes $y = (xe)^{-1} = 61 \pmod{\phi(n)}$, so that $xy = d \pmod{\phi(n)}$ and supplies it to BC.

To decrypt the codetext 54, BC computes

$$\begin{aligned} ((E(a))^x)^y \bmod n &= ((E(a))^x \bmod n)^y \bmod n \\ &= (54^{3087} \bmod 187)^{61} \bmod 187 \\ &= 142^{61} \bmod 187 \\ &= 10 \ . \end{aligned}$$

4.2 Double encryption scheme

We add a second encryption: arbitrarily choose exponent $e' = 99$; then $d' = 139$. The distributed codetext pair is $\langle E(a), E'(a)\rangle = \langle a^e \bmod n, a^{e'} \bmod n\rangle = \langle 54, 65\rangle$.

Customer BC is assigned two identification numbers: $x = 3^2 \cdot 7^3 = 3087$ as before, and $x' = 3^3 \cdot 7^2 = 1323$ generated by considering the characters in the user's name in the opposite order. Suppose the encryptor arbitrarily selects random value $r = 99$; then $1 - r \bmod \phi(n) = 62$. The encryptor computes $y = 119$ and $y' = 126$ to satisfy

$$\begin{aligned} exy &= r \pmod{\phi(n)} \\ e'x'y' &= 1 - r \pmod{\phi(n)} \ , \end{aligned}$$

and supplies y and y' to customer BC.

To decrypt $\langle E(a), E'(a)\rangle$, BC computes, modulo n,

$$\begin{aligned} ((E(a))^x)^y \cdot ((E'(a))^{x'})^{y'} &= (54^{3087})^{119} \cdot (65^{1323})^{126} \\ &= 142^{119} \cdot 10^{126} \\ &= 65 \cdot 144 \\ &= 10 \pmod{187} \ , \end{aligned}$$

revealing the original plaintext.

4.3 Error-correcting coding

In this section we add error-correcting coding to the two-encryption scheme. To keep the example small, we will use only five bits of information from the user name. For user "U", this information is the binary number 10101, as U is the 21st letter.

To error-correcting code 10101, we use the standard Hamming code [Ham50; VvO89, pp. 65–69] for 5 "beef" bits; there are 4 check bits. The matrix is

$$\begin{array}{ccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

where the columns containing a single "1" are the check bits. With the check bits suppressed, the encoding is $\cdot\cdot 1 \cdot 010 \cdot 1$. With check bits set to give an even number of 1s in each row, the encoding is 001101011.

We will Gödel-encode this example one bit at a time; we use the nine primes 3, 7, 11, 13, 17, 19, 23, 29, and 31 (omitting 2 and 5 because they divide $\phi(n) = 160$). The result is $x = 11{\cdot}13{\cdot}19{\cdot}29{\cdot}31 = 2442583$. To compute x', we invert all the bits (including the check bits): $x' = 3 \cdot 7 \cdot 17 \cdot 23 = 8211$.

Let random value $r = 99$ once again. The encryptor computes $y = 111$ and $y' = 78$ and supplies them to user U.

User U decrypts codetext $\langle 54, 65 \rangle$ as before:

$$\begin{aligned} ((E(a))^x)^y \cdot ((E'(a))^{x'})^{y'} &= (54^{2442583})^{111} \cdot (65^{8211})^{78} \\ &= 164^{111} \cdot 109^{78} \\ &= 65 \cdot 144 \\ &= 10 \pmod{187} . \end{aligned}$$

4.4 Revealing xy or $x'y'$

This section extends the previous example, showing that $xy = 271126713$ or $x'y' = 640458$ uniquely identify the user to whom those numbers were issued, just as x and x' do. Determining the username from either product proceeds in three steps.

1. Discard extra factors of xy and $x'y'$. The prime factors of x and x' are among 3, 7, 11, 13, 17, 19, 23, 29, and 31, and each of those appears zero or one times.

$$\begin{aligned} xy &= 3 \cdot 11 \cdot 13 \cdot 19 \cdot 29 \cdot 31 \cdot 37 \\ &\rightarrow 3 \cdot 11 \cdot 13 \cdot 19 \cdot 29 \cdot 31 \end{aligned}$$

$$\begin{aligned} x'y' &= 2 \cdot 3^2 \cdot 7 \cdot 13 \cdot 17 \cdot 23 \\ &\rightarrow 3 \cdot 7 \cdot 13 \cdot 17 \cdot 23 \end{aligned}$$

2. Reverse the Gödel encoding to get a nine-bit string.

$$3 \cdot 11 \cdot 13 \cdot 19 \cdot 29 \cdot 31 \rightarrow 101101011$$

   For x', invert every bit after reversing the Gödel encoding.

$$3 \cdot 7 \cdot 13 \cdot 17 \cdot 23 \rightarrow 110110100 \rightarrow 001001011$$

3. Reverse the error-correcting code to get a five-bit user name.

   Bit pattern 101101011 fails the first of the four equations derived from the Hamming matrix (and passes the others), so the bit corresponding to the column containing a 1 in the first row is incorrect; the nine-bit string should have been 001101011, of which the beef bits are 10101.

   Bit pattern 001001011 fails only the third Hamming equation, so the so the bit corresponding to the column containing a 1 in the third row is incorrect. The nine-bit string is corrected to should have been 001101011, of which the beef bits are 10101.

5 Collusion by multiple users

The Heraclitean encryption scheme described above provides much better key tracking and security than any existing encryption scheme for broadcast media. However, it is not perfect, as it contains a flaw which permits groups of dishonest purchasers to produce new keys.

Suppose purchasers with identification information x_i, x_i' have decryption keys y_i, y_i'. They do not know r_i, the random numbers used by the encryptor in issuing their private keys, but they do know that for each i,

$$ex_iy_i + e'x_i'y_i' = 1 \pmod{\phi(n)} .$$

For arbitrary multipliers a_i,

$$\sum_i a_i e x_i y_i + \sum_i a_i e' x_i' y_i' = \sum_i a_i \pmod{\phi(n)} ,$$

so if $\sum_i a_i = 1$, then

$$e(\sum_i a_i x_i y_i) + e'(\sum_i a_i x_i' y_i') = 1 \pmod{\phi(n)} .$$

The parenthesized expressions, which the conspirators can compute, can be used to decrypt codetext pair $\langle a^e, a^{e'} \rangle$.

This attack does not permit computation of $\phi(n)$ or n, and anyone can determine that the new keys are inauthentic (so the usual installation or decryption programs will not work without modification). The encryptor, who knows the values of all the $xy, x'y'$ pairs, may be able to determine the identities of the conspirators. (The random values r, or the Gödel encodings x, x' of the identifying information, can be chosen to make this identification easier.) It is a shortcoming, but not a fatal one, that the inauthentic keys do not reveal the conspirators to all. It is possible that this scheme can be patched, or that another Heraclitean scheme can be produced which prevents even this sort of attack.

A single legitimate purchaser with knowledge of e and e' (or of $exy + e'x'y' = 1 \pmod{\phi(n)}$) could break the system, but those values cannot be determined from the information given the purchaser. If they could be determined, then the purchaser can find $k \geq 2$ and odd t such that $exy + e'x'y' = 2^k t$. He can choose random s such that $\gcd(s, n)$ and find the smallest integer j such that $s^{2^j t} = 1 \pmod{n}$. Such an integer exists, and can be easily found, because $s^i = 1 \pmod{n}$ for any $i \,|\, \phi(n)$, and $2^k t \,|\, \phi(n)$. $2^{j-1}t$ is a square root of 1, modulo n; with probability $\frac{1}{2}$ it is non-trivial (not $= -1, \bmod n$), so it yields the factorization of n, which the purchaser can distribute with impunity.

6 Practicalities

Heraclitean encryption does not require the use of the RSA cryptosystem; other public-key schemes with similar properties can be just as easily used. In particular, the elliptic curve method [KMOV91] can be substituted for RSA. Its chief advantage over RSA is that elliptic curve systems are more secure for the same key length, or just as secure with shorter, easier-to-communicate keys [Men93].

Heraclitean encryption does not address tracing leaked information; it is a feature of Heraclitean encryption that each decryption produces the same result from a particular codetext, even though the decryption keys differ.

A valid recipient of information can always communicate it to others; the goal of Heraclitean encryption is to prevent there from being an easier (less communication-intensive) way for conspirators to learn the information, even if they all have copies of the codetext. Most encryption schemes permit a decryption key to be anonymously distributed, but Heraclitean encryption prevents this.

The Heraclitean encryption scheme described in this paper doubles the size of the data. The size of the encrypted data can be reduced by using this scheme only for part of the data. If the data is useless unless all of it is recovered, this is an effective way to protect it without unduly increasing data size. However, this decreases security, because only those parts of the data (and a short key or program for decrypting the weakly-protected parts) need be communicated.

No encryption scheme exists in a vacuum, insulated from social pressures. An application of Heraclitean encryption to software distributed on CD-ROM, like that described in the introduction, could be circumvented without breaking the encryption. For instance, a false identity or stolen credit card could be used, or the credit card could be canceled immediately after the purchase. The software company can fight back, also in the realm of psychology rather than mathematics. For instance, multiple CD-ROMs can be printed, differing in their encryption exponents. A pirate who types a decryption key that is valid for a different CD-ROM batch can be directed to call the manufacturer for help (and to incriminate himself). Or, a cash reward and amnesty could be offered for any member of a piracy ring turning state's evidence. Success is achieved by making beating the system as hard as other sorts of piracy, and more difficult than the value of the software

warrants. For mass-market software, the scheme described is feasible.

References

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, April 1950.

[Her] Heraclitus. On the universe. Fragment 41. Quoted in *Bartlett's Familiar Quotations*, published by Little, Brown, and Company, 1980.

[KMOV91] Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto, and Scott A. Vanstone. New public-key schemes based on elliptic curves over the ring $\mathbf{Z}_n$. In Joan Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91 Proceedings*, number 576 in Lecture Notes in Computer Science, pages 252–266. Springer-Verlag, 1991.

[LP81] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall Software Series. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[Men93] Alfred J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Communications and information theory. Kluwer Academic Publishers, Boston, MA, 1993.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[VvO89] S. A. Vanstone and P. C. van Oorschot. *An Introduction to Error-Correcting Codes with Applications.* Kluwer, Boston, 1989.