Proc. 1983 Symp. on Security + Privacy

THE MANY-TIME PAD: THEME AND VARIATIONS

Dorothy E. Denning

Computer Sciences Dept.
Purdue University
W. Lafayette, IN 47907

Prelude

The *many-time pad* is a method of subverting the security controls of a system to obtain data that is not directly accessible (e.g., because the data is confidential, classified, or otherwise deemed sensitive). It is the antithesis of the *one-time pad*, the only theoretically unbreakable cipher, in two respects: 1) whereas the one-time pad is a method of protection, the many-time pad is a method of attack; and 2) whereas the one-time pad is used just once, the many-time pad is reusable. Also, whereas the interpretation of "pad" in the one-time pad comes from a "pad of paper", its interpretation in the many-time pad comes from "stuffing".

What makes the many-time pad attack interesting is that it arises in three different contexts: cryptographic systems, where digital signatures can be forged or messages decrypted; statistical databases, where trackers can be used to obtain confidential data; and programming systems, where Trojan Horses can be planted in programs to leak sensitive input data. We shall first describe the basic structure of the attack and countermeasures for foiling it. We shall then show how these three seemingly unrelated security threats are variations of a common theme.

Theme

The general setting of the attack is a system that accepts queries from a user for the value of a function f on an input X supplied by a user. For each query, the system either supplies the result $f(X)$ or withholds it for security reasons. The objective of the attack is to obtain $f(X)$ when it is withheld. The method of the attack involves padding X to make the query look innocent, thereby disguising the user's true intent to obtain $f(X)$. The padding is craftily arranged so that its effect on the output is easily removed.

The first step in the attack is to find an input P (the *pad*) for which the system will return $f(P)$. Next, X is padded with P by computing $(P \circ X)$, where "$\circ$" is a binary operator over the input domain, and $f(P \circ X)$ is requested. Assuming the system returns $f(P \circ X)$, the effect of the padding is removed by computing $f(P)^{-1} \bullet f(P \circ X)$, where $\bullet$ is a binary operator over the output domain and $f(P)^{-1}$ is the inverse of $f(P)$ with respect to $\bullet$. The result of the computation will be $f(X)$ if two conditions are met: 1) the input and output domains with their respective binary operators form groups so that the operators are associative and inverses can be computed; and 2) the operator f is a *homomorphism* from the input group to the output group; that is,

$$f(P \circ X) = f(P) \bullet f(X) . \tag{1}$$

As we shall see, the attack may also work under somewhat weaker conditions. The steps of the attack are summarized in the following:

The Many-Time Pad for Obtaining $f(X)$.

1. Find a pad P for which the system will return $f(P)$.
2. Pad X with P and request $f(P \circ X)$.
3. If successful, compute

$$\begin{aligned} f(P)^{-1} \bullet f(P \circ X) &= f(P)^{-1} \bullet (f(P) \bullet f(X)) \\ &= (f(P)^{-1} \bullet f(P)) \bullet f(X) = f(X) . \end{aligned}$$

The computation in step 3 is summarized by the following rule of inference:

$$f(P)^{-1} \bullet f(P \circ X) = f(X) , \tag{2}$$

which is derived from the underlying homomorphic structure. We shall refer to Eq. (2) as the *padding rule of inference*.

Note that once a pad P is picked and $f(P)$ obtained, the same pad can be reused with different values for X, eliminating the need to repeat step 1.

There are two basic strategies for foiling the many-time pad attack: restriction and perturbation. With restriction, the system is much more selective in the queries it answers. In particular, it withholds $f(P)$ or $f(P \circ X)$ whenever $f(X)$ is sensitive. This may be easier said than done, however, if the queries are nonsensitive when considered separately. If the restriction criterion is too permissive, it may not thwart the attack; if it is too conservative, it can cause considerable information loss.

With perturbation, the system introduces noise into the computation of f. Letting f' denotes the perturbed function, f' must either 1) destroy the underlying homomorphic structure of the system so that $f'(P \circ X) \neq f'(P) \bullet f'(X)$, in which case the padding rule of inference fails; or 2) introduce enough noise that an estimate $f'(X)$ obtained by the padding rule is not considered sensitive. We will see examples of both strategies in countermeasures for variations of the attack.

where h is one-way public hashing function, computed over the entire message, and I is a random seed used to initialize h. The seed is picked by the user to introduce uncertainty into the message signed. It is concatenated to the result $h(X, I)$, and the entire block transformed by f.

A signature S on an alleged message X is validated in three steps:
1) compute $f^{-1}(S) = (h(X, I)\ I)$; 2) compute $h(X, I)$ using the public function h, the alleged message X, and the seed I obtained in step 1; and 3) compare the $h(X, I)$ obtained in step 1 with that computed in step 2. The message X is transmitted either as cleartext (if secrecy is not needed), as ciphertext encrypted using the receiver's public key, or as ciphertext encrypted using a secret key shared by the sender and receiver (if a conventional system such as the DES is used for message secrecy, with the public-key system reserved for signatures and key exchange).

One possibility for the function h, suggested by Wolfgang Bitzer*, is given by the following: First, the message X is broken into 56-bit blocks $X_1, \ldots, X_r$. Letting E_{X_i} denote the DES enciphering algorithm keyed to block X_i, $h(X, I) = Z_{r+1}$, where

$$Z_{i+1} = E_{Z_i' \oplus X_i}(Z_i) \quad (1 \le i \le r)$$

$$Z_1 = I,$$

Z_i' consists of 56 bits selected from Z_i and and $\oplus$ denotes exclusive or.

The Davies-Price protocol with the DES-based hashing function foils the many-time pad attack by destroying the homomorphic structure of the cryptographic system. Specifically, $h(P \circ X, I) \neq h(P, I) \circ h(X, I)$, whence $f(h(P \circ X, I)\ I) \neq f(h(P, I)\ I) \bullet f(h(X, I)\ I)$ is almost certain to hold. Thus, the padding rule of inference cannot be used to factor out the pad P from a signature on $(P \circ X)$ even if the same seed is used. The Davies-Price signature protocol has several other advantages as well [6, 7, 4].

Variations for Statistical Databases

A statistical database returns statistics computed from groups of records having common attribute values. The goal is to provide as much information as possible without compromising the privacy of the individuals (or organizations) represented therein. To achieve this goal, the database must suppress sensitive statistics about single individuals as well as other statistics that could lead to their inference.

Statistical databases can be vulnerable to a many-time pad attack if countermeasures to thwart such attacks are not in place. Such attacks are called ***tracker attacks*** [8, 9, 10, 11] because they allow a user to "track down" information about identifiable individuals in the database. A "tracker" is just a type of pad.

A statistic is given by $S = f(X)$, where X is a logical formula and f is a statistical function computed from the set of records satisfying X (called the *query set* of X). An example of a statistic is S = SAL.SUM(SEX=FEMALE AND AGE=37), which gives the sum of the salaries of all females age 37. We are particularly concerned with additive statistics [12], since these are most vulnerable to tracker attacks. An ***additive statistic*** is one that satisfies the formula

$$f(P \text{ OR } X) = f(P) + f(X), \tag{3}$$

when P and X are formula having disjoint sets. Counts, sums, and higher-order moments [13] are additive.

Because Eq. (3) does not hold for all formula P and X, the statistical function f is not a true homomorphism. Nevertheless, because Eq. (3) holds when the sets identified by P and X are disjoint and inverses can be computed in the output domain, the padding rule of inference can be applied to obtain a sensitive statistic $f(X)$:

Compute Sensitive Statistic $S = f(X)$.

1. Find a tracker (pad) P such that the query sets for P and X are disjoint and such that the database system will release the statistic $f(P)$.
2. Pad X with P and request the statistic $f(P \text{ OR } X)$.
3. If successful, compute $-f(P) + f(P \text{ OR } X) = f(X)$.

The following illustrates

Example 1. Let

$S = f(X)$ = SAL.SUM(SEX=FEMALE AND AGE=37).

Suppose that X identifies a single individual, whence S would be withheld to protect her salary. Using a many-time pad, the sensitive statistic S can be inferred from statistics over much larger query sets as follows:

1. Let P = (SEX=FEMALE AND AGE≠37) and obtain SAL.SUM(SEX=FEMALE AND AGE≠37) = 2321100.
2. Compute $(P \text{ OR } X)$ = (SEX=FEMALE) and obtain SAL.SUM(SEX=FEMALE) = 2352700.
3. Compute
   - SAL.SUM(SEX=FEMALE AND AGE≠37)
   + SAL.SUM(SEX=FEMALE) = 31600.

Even when the sets P and X overlap, padding can be used to obtain $f(X)$. For example, using a "general tracker" [8],

$$-f(P) + f(P \text{ OR } X)$$
$$-f(\text{NOT } P) + f(\text{NOT } P \text{ OR } X) = f(X).$$

Note that $f(X)$ cannot generally be obtained from

$$-f(P) + f(P \text{ OR } X) + f(P \text{ AND } X) = f(X)$$

because $f(P \text{ AND } X)$ will be sensitive when $f(X)$ is sensitive.

To protect against the many-time pad as well as other methods of attack, statistical databases use the two basic controls, restriction and perturbation.

Restriction techniques aim to prevent inference of sensitive statistics by withholding additional nonsensitive statistics. To see how restriction can foil the many-time pad attack, consider the queries in Example 1. The desired salary sum is for a group defined by two attribute values, SEX and AGE. If we consider the set of all such statistics, they form a 2-dimensional table (2-table for short) broken down by SEX and AGE, with 2 rows for SEX and, say, 100 columns for AGE (see Figure 1). The entire table contains 200 cells. If we now look at

* Bitzer's function was brought to my attention by Donald Davies.

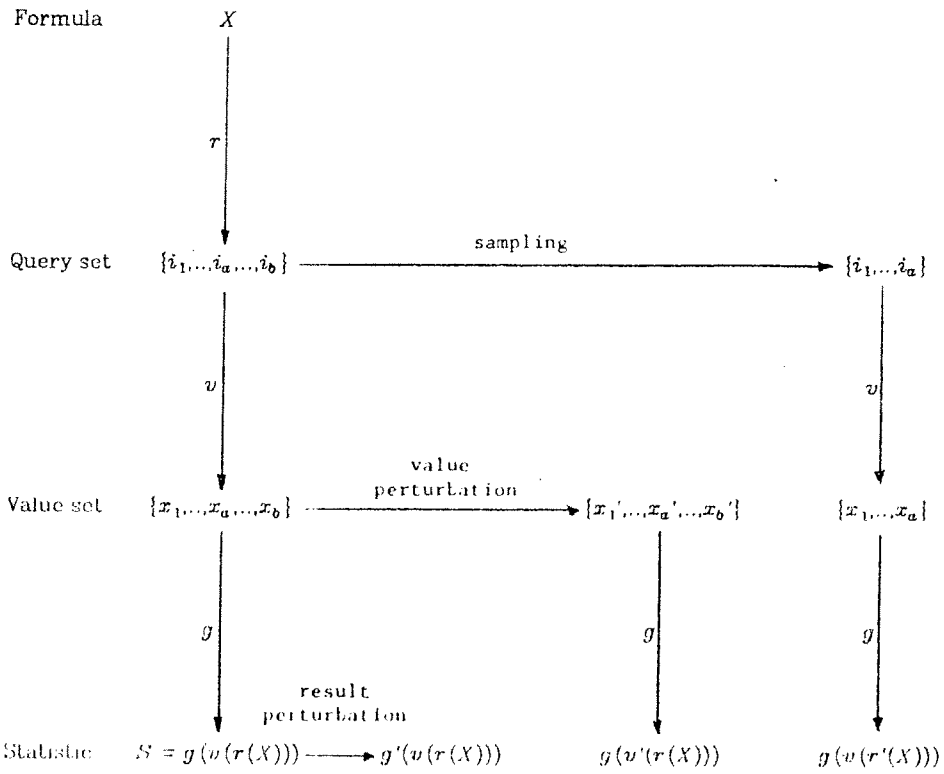

Figure 2. Perturbation techniques.

Techniques applied after r modify the query set $I = r(X)$. An example is *Random Sample Queries* (RSQs) [20], which computes the statistic from a random sample of the records in I (see Figure 2). For notational convenience, Figure 2 shows this subset to be the first a elements of the set I, where $a \leq b$. RSQs have the property of destroying the level 1 homomorphic structure. In particular, the sample for $(P \text{ OR } X)$ can be expected to differ from the union of the samples for P and X; that is,

$$r'(P \text{ OR } X) \neq r'(P) \cup r'(X) .$$

This property foils most attempts to isolate records in tracker attacks; in particular, an attacker cannot expect to get a good estimate of a sensitive statistic from the padding rule of inference employed in step 3 of the attack.

Techniques applied after v perturb each data value in the value set (see Figure 2). Examples are given in [21, 22, 19]. If the errors introduced into each data value are a function of the query set, then the errors introduced into the set $(P \text{ OR } X)$ can be expected to differ from those introduced into the sets P and X, thereby destroying the homomorphic structure at level 2; that is,

$$v'(r(P \text{ OR } X)) \neq \{v'(r(P)) : v'(r(X))\} .$$

If, however, the errors are held fixed for each record, then this structure, as well as consistency, is preserved, though there are certain security risks associated with preserving consistency [19].

Techniques applied after g is computed simply perturb the result (see Figure 2). Examples are systematic and random rounding, systematic and random ranges, and controlled rounding [23, 24, 25, 26, 19]. Random ranges seem to be more acceptable to users than the other strategies, and also have the advantages of high security and no bias [19].

Perturbation techniques for statistical databases should maintain some constancy; that is, repeated queries for the same statistic should, as much as possible, return the same response. This is needed to prevent averaging attacks whereby a true statistic is obtained by averaging different responses to equivalent queries. Because the same query set can be expressed in many different ways, constancy requires either making the perturbation dependent on the composition of the query set, or else adopting a restricted syntax for queries that allows easy reduction to a normal form and reduces the number of ways of obtaining estimates [27].

Constancy is not achieved by the hashing/signature function because the user picks a new seed I each time a message is signed. Indeed, constancy should be avoided in this context: if a constant seed were used for all messages, an attacker might be able to forge a signature on a message X by finding a variation of X with the same hash value as a message the victim is willing to sign.

Now, if the perceived running times of the program is, say, 25, then x could be 1, 2, or 3, so the perturbation seems to have foiled the attack. Unfortunately, such perturbation techniques can be subverted -- in this case, by changing the loop of the program to run from 1 to 3^x, giving

x	$t(P, 3^x Q)$	$t'(P, 3^x Q)$
1	25	25-50
2	55	55-110
3	145	145-290
4	415	415-830

Since the time intervals do not overlap, the value of x can now be inferred from the perturbed running time. If the error is magnified even further, the control can be subverted with a larger exponent.

We can attempt to foil the attack by requiring that all programs specify in advance their resource requirements, assigning these resources to the program (even if they are not needed), and returning the results (even if incomplete) at precisely the time specified. With this strategy, nothing can be deduced from running time that was not known beforehand (actually one bit of information can be deduced from whether a program successfully completes). The program in step 2 of the attack, for example, would have a fixed running time for all values of x, whence nothing could be deduced about x in step 3. Of course, the impact of this control on system performance is considerable.

The above approach does not prevent a user from specifying a small constant running time for the program in step 2, and successively rerunning the program until it terminates within the allotted time, thereby revealing x. This can be prevented only if the system keeps a log of all abnormal terminations due to timeouts, and uses the log to detect unauthorized activity.

Coda

The many-time pad threatens data security in at least three different contexts: cryptographic systems, statistical databases, and programming systems. In all three contexts, padding is used to conceal a query for unauthorized data. The effect of the padding is removed through a rule of inference derived from an underlying homomorphic structure. Countermeasures for the attack involve restriction techniques, which limit the availability of information to the attacker, and perturbation techniques, which often destroy the homomorphic structure.

Of course, the many-time pad is not the only threat to data security. For example, cryptographic systems are also threatened by cryptanalysis, traffic analysis, and compromise or destruction of keys. Statistical databases are threatened by direct disclosure of confidential data through access control circumvention. Programming systems are threatened by Trojan horses that obtain unauthorized access to data, and by resource hungry programs that prevent other programs from being serviced. It is difficult to compare the seriousness of the many-time pad with these other forms of attack without knowing how the system is being used and what security controls are in place. But without adequate controls, the many-time pad could undermine the security of a system.

Acknowledgements

Thanks to Peter Denning for suggesting the term many-time pad and for many helpful comments, and to Bob Blakley, Jon Millen, and Jan Schlörer for useful suggestions. This research was supported in part by NSF Grant MCS80-15484.

References

1. Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM* 21(2) pp. 120-126 (Feb. 1978).
2. Davida, G. I., "Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem," TR-CS-82-2, Dept. of EE and CS, Univ. of Wisconsin, Milwaukee, WI (October, 1982).
3. DeMillo, R. and Merritt, M. J., "Chosen Signature Analysis of Public-Key Cryptosystems," Technical Memorandum, Georgia Tech. (Oct. 1982).
4. Denning, D. E., "Signature Protocols for RSA and Other Public-Key Cryptosystems," CSD-TR-419, Computer Sciences Dept., Purdue Univ. (Nov. 1982).
5. Shamir, A., "A Fast Signature Scheme," MIT/LCS/TM-107, MIT Lab. for Comp. Sci., Cambridge, Mass. (July 1978).
6. Davies, D. W. and Price, W. L., "The Application of Digital Signatures Based on Public Key Cryptosystems," NPL Report DNACS 39/80, National Physical Lab., Teddington, Middlesex, England (Dec. 1980).
7. Denning, D. E., "Protecting Public Keys and Signature Keys," *IEEE Computer* 16(2) pp. 27-35 (Feb. 1983).
8. Denning, D. E., Denning, P. J., and Schwartz, M. D., "The Tracker: A Threat to Statistical Database Security," *ACM Trans. on Database Syst.* 4(1) pp. 76-96 (Mar. 1979).
9. Denning, D. E. and Schlörer, J., "A Fast Procedure for Finding a Tracker in a Statistical Database," *ACM Trans. on Database Syst.* 5(1) pp. 88-102 (Mar. 1980).
10. Schlörer, J., "Identification and Retrieval of Personal Records from a Statistical Data Bank," *Methods Inf. Med.* 14(1) pp. 7-13 (Jan. 1975).
11. Schlörer, J., "Disclosure from Statistical Databases: Quantitative Aspects of Trackers," *ACM Trans. on Database Syst.* 5(4) pp. 467-492 (Dec. 1980).
12. Denning, D. E., Schlörer, J., and Wehrle, E., "Memoryless Inference Controls for Statistical Databases," Computer Sciences Dept., Purdue Univ. (1982).
13. Dalenius, T. and Denning, D., "A Hybrid Scheme for Statistical Release," *Statistisk Tidskrift*, (2) pp. 97-102 (1982).
14. Cox, L. H., "Suppression Methodology and Statistical Disclosure Control," *J. Amer. Stat. Assoc.* 75(370) pp. 377-385 (June 1980).
15. Sande, G., "Towards Automated Disclosure Analysis for Establishment Based Statistics," Statistics Canada (1977).