Dense Probabilistic Encryption

Josh Benaloh
Clarkson University

Abstract

This paper describes a method of *dense* probabilistic encryption. Previous probabilistic encryption methods require large numbers of random bits and produce large amounts of ciphertext for the encryption of each bit of plaintext. This paper develops a method of probabilistic encryption in which the ratio of ciphertext text size to plaintext size and the proportion of random bits to plaintext can both be made arbitrarily close to one. The methods described here have applications which are not in any apparent way possible with previous methods. These applications include simple and efficient protocols for non-interactive verifiable secret sharing and a method for conducting practical and verifiable secret-ballot elections.

1 Introduction

In 1984, Goldwasser and Micali ([GoMi84]) introduced the notion of *probabilistic encryption.* A probabilistic encryption method allows one to encrypt a fixed value in many different ways. Thus, even when given the encryption of a value and details of the encryption mechanism (including any encryption key), it is not necessarily possible for an adversary to determine whether or not a given ciphertext represents the encryption of a particular value.

Goldwasser and Micali develop a bit encryption function based on the number theoretic problem of quadratic residuosity. Their method has many useful properties, but there is one major drawback: for a given security parameter N, the probabilistic encryption of each bit is N bits long, requires N random bits, and uses several operations on N bit integers.

This work describes a dense method of probabilistic encryption which, unlike the method of Goldwasser and Micali, is capable of encrypting more than one bit at a time. For any given k and security parameter N, this new method allows the encryption of k bits of information into an $N + k$ bit ciphertext using $N + k$ random bits and operations on $N+k$ bit integers. Thus, for any desired security parameter N, the ratio of plaintext size to ciphertext size (as well as to random bits required or to the size of the integers computed upon) can be made arbitrarily close to one.

There are also some applications where one bit at a time probabilistic encryption is unsuitable regardless of efficiency. This paper describes two such applications – non-interactive verifiable secret sharing and a method for obtaining verifiable secret-ballot elections – in which the dense probabilistic encryption method described here can be used while there is no apparent way of developing similar solutions with bitwise probabilistic encryption.

2 The Encryption Method

This section will show how to generate "one-to-many" functions (or *randomized* functions) E_r for any odd integer r with the following basic properties.

- Given a message $M \in \mathbb{Z}_r = \{0, 1, 2, \ldots, r-1\}$, it is computationally easy to for any participant to form *an* encryption $z \in E_r(M)$.
- The decryption of any $z \in E_r(M)$ is unique – that is, if $M_1, M_2 \in \mathbb{Z}_r$ with $M_1 \neq M_2$, then $E_r(M_1) \cap E_r(M_2) = \emptyset$; and this unique decryption can be computed by the creator of E_r.
- Under a suitable cryptographic assumption it is "infeasible" to compute M (or even gain so much as an inverse polynomial advantage at computing any predicate on M) given simply the details of the randomized function E_r and an encryption $z \in E_r(M)$.

In addition to these properties, several other useful properties will be achieved.

- Given a message $M \in \mathbb{Z}_r$ any participant can generate a $z \in E_r(M)$ together with a certificate u which can be used to prove to any other participant(s) that $z \in E_r(M)$.
- Given an encryption $z \in E_r(M)$, it is possible for the creator of E_r to produce a certificate u which is uniformly selected from the set of all possible user certificates and can, likewise, be used to prove that $z \in E_r(M)$.
- There are easily (and universally) computable functions "$\otimes$" and "$\oslash$" which have the property that whenever $z_1 \in E_r(M_1)$ and $z_2 \in E_r(M_2)$ it is the case that $(z_1 \otimes z_2) \in E_r((M_1 + M_2) \bmod r)$ and $(z_1 \oslash z_2) \in E_r((M_1 - M_2) \bmod r)$.

2.1 Encryption

A randomized encryption function E_r satisfying the above properties can be developed as follows.

1. Select two "large" primes p and q such that r divides $(p-1)$, r and $(p-1)/r$ are relatively prime, and r and $(q-1)$ are relatively prime. (Such primes are easy to generate by searching among appropriate arithmetic sequences.) Form $n = pq$.
2. Select $y \in \mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$ such that $y^{(p-1)(q-1)/r} \bmod n \neq 1$.
3. Reveal n and y. The randomized encryption function E_r is defined by the set $E_r(M) = \{y^M u^r \bmod n : u \in \mathbb{Z}_n^*\}$.

It is now a trivial matter for a user given a message $M \in \mathbb{Z}_r$ and randomized function E_r to (by randomly and uniformly selecting $u \in \mathbb{Z}_n^*$) generate a uniform element $z = (y^M u^r \bmod n) \in E_r(M)$. Furthermore, this u serves as a certificate to prove that $z \in E_r(M)$.

To see that decryptions are unique, observe that $y^{M_1} u_1^r \bmod n = y^{M_2} u_2^r \bmod n$ implies that $y^{M_1 - M_2} \bmod n = (u_2 u_1^{-1})^r \bmod n$. By the construction of y, this, in turn, implies that $M_1 \bmod r = M_2 \bmod r$. It then follows immediately that the sets $E_r(0), E_r(1), E_r(2), \ldots, E_r(r-1)$ form a partition of $\mathbb{Z}_n^*$, and this gives the unique decryption property. Also, the two functions given by $z_1 \otimes z_2 = (z_1 \cdot z_2) \bmod n$ and $z_1 \otimes z_2 = (z_1 \cdot z_2^{-1}) \bmod n$ can easily be seen to satisfy the homomorphic properties described above.

In contrast, the original Goldwasser-Micali probabilistic encryption is essentially the special case of this method in which $r = 2$. However, since it is impossible for $r = 2$ to be relatively prime to $q - 1$ when q is a large prime, the Goldwasser-Micali system must restrict consideration to those elements of $\mathbb{Z}_n^*$ with a Jacobi symbol of $+1$. There is no need for such a restriction in the dense system described here.

2.2 Decryption

Security of decryption is based on the cryptographic assumption that deciding *higher residuosity* is computationally difficult: *given z, r, and n of unknown factorization, there is no known polynomial time algorithm to determine whether or not there exists an x such that $z = x^r \bmod n$.*

In the case where $n = pq$ is of the form described above and the prime factors p and q are known, the process of deciding higher residuosity is efficiently computable by the following simple rule:

$$z \in E_r(0) \qquad \text{if and only if} \qquad z^{(p-1)(q-1)/r} \bmod n = 1.$$

Thus, if r is small, one can simply decrypt a message z by determining (exhaustively) the smallest non-negative integer m such that $(y^{-m}z \bmod n) \in E_r(0)$.

This process can be further accelerated by pre-processing. For each $M \in \mathbb{Z}_r$, a canonical value T_M can be computed as

$$T_M = y^{M(p-1)(q-1)/r} \bmod n.$$

It can be shown that for every $z \in E_r(M)$, it is true that $z^{(p-1)(q-1)/r} \bmod n = T_M$. Thus, the r (distinct) values $T_0, T_1, \ldots, T_{r-1}$ can be pre-computed, and any encrypted value z can be decrypted by a table look-up on the value $z^{(p-1)(q-1)/r} \bmod n$.

If r is of moderate size, a combination of the two previous methods can be used to bring the storage, pre-computation time, and decryption time all to $\mathcal{O}(\sqrt{r})$. The idea (sometimes known as "big-step little-step" is to pre-compute T_M for $M \approx k\sqrt{r}$ as k ranges from 1 to $\sqrt{r}$. These values serve as milestones which are only about $\sqrt{r}$ steps apart. Given a z of unknown decryption, one can find the smallest non-negative integer m for which the T_M corresponding to $y^{-m}z$ has been pre-computed. This m is bounded above by $\sqrt{r}$ and can be regarded as an offset from the pre-computed decryption value T_M. The decryption of such a z is therefore the value $M + m$.

Finally, even if r is large, decryption is efficient provided that r contains no large prime factors. An extreme case in which decryption is very efficient is when r is of the form $r = 3^k$ for some positive integer k. In this case, the decryption of a value z can be computed in ternary notation. First, the low order "trit" t_k of the decryption of z is the unique value $t_k \in \{0, 1, 2\}$ such that $(y^{-t_k}z)^{(p-1)(q-1)/3} \bmod n = 1$. Once t_k has been computed, the next-to-last trit $t_{k-1} \in \{0, 1, 2\}$ is computed as the unique value such that $(y^{-t_k - 3t_{k-1}}z)^{(p-1)(q-1)/3^2} \bmod n = 1$. Next, $t_{k-2} \in \{0, 1, 2\}$ is computed as the unique value such that $(y^{-t_k - 3t_{k-1} - 3^2 t_{k-2}}z)^{(p-1)(q-1)/3^3} \bmod n = 1$. This process is continued until the ternary representation $\langle t_1, t_2, \ldots, t_k \rangle$ of the decryption of z is computed.

3 Some Applications

Besides the advantages of greater density in probabilistic encryption, there are some tasks which can be performed with the methods described here which cannot, in any apparent way, be done by any other means whatsoever.

3.1 Verifiable Secret Sharing

The notion of secret sharing was introduced by Shamir in [Sham79]. Shamir defines *threshold schemes* to be methods of dividing a secret value s into *shares* such that

(1) Any subset of shares which exceeds a predetermined size is sufficient to reconstruct the secret.

(2) Any smaller subset of shares gives no information (in an information theoretic sense) about the secret.

Shamir described a method of secret sharing based on polynomial interpolation and evaluation.

In 1985, Chor, Goldwasser, Micali, and Awerbach ([CGMA85]) described the problem of *verifiable* secret sharing. The problem here is to develop a protocol for secret sharing such that when it is complete each shareholder is confident that its share is meaningful. (Note that a dishonest secret sharer could give some shareholders worthless information rather than actual shares.)

Chor, Goldwasser, Micali, and Awerbach give a protocol which achieves verifiable secret sharing. However, their method is exponential in the number of shareholders.

The application of the encryption method described in this paper to the problem of verifiable secret sharing was first given in [Bena86] in which interactive proof techniques are also required. Feldman ([Feld87]), Ben-Or, Goldwasser, and Wigderson ([BGW88]), Rabin ([Rabi88]), and Rabin and Ben-Or ([RaBO89]) later expanded upon this approach.

The basic technique used in all of these methods is to perform computations on shares of secrets without first combining the shares to form the secrets. This is easily possible if the shares are encrypted using the dense probabilistic encryption method given. Computations on shares can be performed using the homomorphism properties described in the previous section.

One of the simplest of these verifiable secret sharing methods is formed by combining the encryption method described herein with Shamir's polynomial based secret sharing ([Sham79]) and the ideas of Feldman's non-interactive verifiable secret sharing ([Feld87]).

To distribute shares of a secret value $s \in \mathbb{Z}_r$ (r must be prime) to m participants such that any k of the m can determine the secret value, a randomized encryption function E_r is formed by the "dealer" of the secret. The dealer then randomly selects values $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_r$ and forms the polynomial

$$P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_2x^2 + a_1x + a_0$$

where the constant coefficient a_0 is given by $a_0 = s$. The dealer then forms m shares $s_i = P(i) \bmod n$ for $1 \leq i \leq m$ and privately distributes each share s_i to the i^{th} shareholder. Next, the dealer computes encryptions $z_j \in E_r(a_j)$ of the coefficients a_j for $0 \leq j < k$ and publicly reveals these encryptions. It is now possible for any and all participants to (by using the homomorphism properties) compute

$$w_i = \left(\prod_{j=0}^{k-1} z_j^{i^j} \right) \bmod n.$$

Since $z_j \in E_r(a_j)$, the additive homomorphism property of the cryptosystem implies that $(z_j)^\alpha \in E_r(a_j \alpha)$ for any scalar α. In particular, $z_j^{i^j} \in E_r(a_j i^j)$. Since the share $s_i = P(i) \bmod n = \left(\prod_{j=0}^{k-1} a_j i^j \right) \bmod n$, the additive homomorphism property also implies that each $w_i \in E_r(s_i)$. Thus, each w_i is a publicly computable encryption of the share s_i. The dealer can then privately distribute to the i^{th} shareholder a certificate u_i to prove that $w_i \in E_r(s_i)$. It is thereby impossible for the dealer to convince a shareholder that its share is legitimate when it is not. The secret value s can now be reconstructed at any subsequent time by any k of the shareholders. They need only pool their shares to interpolate the polynomial $P(x)$. The constant coefficient $P(0)$ is, by definition, the secret. This method has the added advantage that since shareholders have certificates of their shares, they cannot convincingly lie to each other about the values of their shares. Thus, dishonest shareholders cannot disrupt the reconstruction of the secret.

One final observation is that the dealer need not even be the creator of the randomized encryption function E_r. Careful examination of the process shows that the share certificates u_i, can be computed directly and simply from the random values the dealer chose to encrypt the coefficients. Hence, the dealer need not even be able to decrypt E_r to engage in this protocol.

It should be noted here that it is simply not possible to use ordinary Goldwasser-Micali probabilistic bit encryption for this application. With Shamir's method of secret sharing, the space from which secret and share values are chosen must be of a size which is prime and *greater than the number of shareholders.* Since probabilistic encryption methods will be used to encrypt shares, the Goldwasser-Micali bit encryption method is inadequate. (Note that segmenting share values bit by bit does not work since the Shamir scheme can only be applied when the number of shareholders is smaller than the secret/share space — which in this case is 2.)

3.2 Verifiable Secret-Ballot Elections

The problem of verifiable secret-ballot elections is defined in [Bena87], and a solution is presented there which depends strongly on the dense probabilistic encryption described here. The general problem is quite complex and its solution requires many techniques which are not addressed here. Instead of trying to present here a complete solution to the verifiable secret-ballot election problem, an overview will be given which demonstrates the use of dense probabilistic encryption.

The oversimplified scheme presented here has many omissions and should not be read as a claim of a secure method of holding verifiable secret-ballot elections. The reader is referred to [Bena87] for a complete treatment.

The scheme described in this section is centralized. A central *government* is assumed to exist. The government prepares a dense probabilistic encryption function E_r as described herein with r greater than the number of eligible voters and publicly reveals E_r.

Each voter selects either a random $v_i \in E_r(0)$ to denote a "no vote" or a random $v_i \in E_r(1)$ to denotes a "yes vote". Each voter then publicly releases its v_i. By the homomorphism properties, $W = (\prod v_i) \bmod n$ is an encryption of the *sum* of the unencrypted values. In this case, therefore, W is an encryption of the total number of yes votes cast by voters. Thus, the central government need only decrypt this one value W to determine the tally of the election; and by providing a certificate u of this decryption, the government can prove to all observers that the claimed tally is accurate.

There are, of course, many problems with this scheme. But this is the fundamental idea used in [CoFi85], [BeYu86], [Cohe86], [Bena87], and [BeTu94] to enable a variety of practical verifiable election schemes.

4 Conclusions

This paper has described a method of dense probabilistic encryption which has many similarities to, but many advantages over, the original method of probabilistic encryption introduced by Goldwasser and Micali. These advantages also apply relative to all previous methods of probabilistic encryption.

Many variations of this method are possible depending on the users' willingness to depend upon stronger assumptions in exchange for more efficient decryption. Applications of this method have also been given in which traditional probabilistic encryption methods are not just less efficient, but are instead unusable.

Acknowledgements

The author would like to express many thanks to Don Coppersmith, Mike Fischer, Neil Immerman, Joe Kilian, Dwight Tuinstra, and Moti Yung who offered helpful comments and suggestions regarding this work.

References

[Bena86] **Benaloh, J.** "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[Bena87] **Benaloh, J.** "Verifiable Secret-Ballot Elections." Ph.D. Thesis presented at *Yale University*, New Haven, CT (Dec. 1987). (Available as *TR-561, Yale University, Department of Computer Science*, New Haven, CT (Sep. 1987).)

[BeTu94] **Benaloh, J.** and **Tuinstra, D.** "Receipt-Free Secret-Ballot Elections." To appear in *Proc. 26th ACM Symp. on Theory of Computing*, to be held in Montreal, PQ (May 1994).

[BeYu86] **Benaloh, J.** and **Yung, M.** "Distributing the Power of a Government to Enhance the Privacy of Voters." *Proc. 5th ACM Symp. on Principles of Distributed Computing*, Calgary, AB (Aug. 1986), 52–62.

[BGW88] **Ben-Or, M.**, **Goldwasser, S.**, and **Wigderson, A.** "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation." *Proc. 20th ACM Symp. on Theory of Computing*, Chicago, IL (May 1987), 1–10.

[CGMA85] **Chor, B.**, **Goldwasser, S.**, **Micali, S.**, and **Awerbuch, B.** "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults." *Proc. 26th IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 383–395.

[CoFi85] **Cohen, J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proc. 26th IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[Cohe86] **Cohen, J.** "Improving Privacy in Cryptographic Elections." *TR-454, Yale University, Department of Computer Science*, New Haven, CT (Feb. 1986).

[Feld87] **Feldman, P.** "A Practical Scheme for Non-interactive Verifiable Secret Sharing." *Proc. 28th IEEE Symp. on Foundations of Computer Science*, Los Angeles, CA (Oct. 1987), 427–437.

[GoMi84] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption." *J. Comp. Sys. Sci. 28*, (1984), 270–299.

[Rabi88] **Rabin, T.** "Robust Sharing of Secrets when the Dealer is Honest or Cheating." *TR 88-1, Hebrew University, Department of Computer Science*, Jerusalem, ISRAEL (July 1988).

[RaBO89] **Rabin, T.** and **Ben-Or, M.** "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority." *Proc.* 21st *ACM Symp. on Theory of Computing*, Seattle, WA (May 1989), 73–85.

[Sham79] **Shamir, A.** "How to Share a Secret." *Comm. ACM 22*, 11 (Nov. 1979), 612–613.