This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Secret Sharing in the Encrypted Domain

Bin Zhao and Edward J. Delp
Video and Image Processing Laboratory
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana, USA

***Abstract*—Secret sharing refers to dividing a secret into pieces or shares and allocating the shares among a group of participants. The secret can be reconstructed only when a sufficient number of shares are combined. To protect each share during secret reconstruction, it is desirable to reconstruct the secret directly from the encrypted shares. A composite algorithm using binary representation and precomputation is developed for efficient exponentiation of encrypted data. A new scheme of secret sharing in the encrypted domain is proposed that makes use of the efficient exponentiation algorithm. Experimental results verify the effectiveness of the efficient exponentiation algorithm and the scheme of secret sharing in the encrypted domain.**

***Index Terms*—Secret sharing, encrypted domain processing, information security, probabilistic encryption, homomorphic encryption, public key cryptosystem.**

I. Introduction

Secret sharing, firstly introduced by Shamir [1] and Blakley [2], is a scheme of sharing a secret among a group of participants. In this scenario, a dealer wants to share a secret, and he gives each participant a "share" of the secret that can be used to reconstruct the secret later. Only certain subsets of participants satisfying a threshold access structure can reconstruct the secret successfully, while any subsets of participants not satisfying the threshold access structure cannot learn anything about the secret. Stinson [3] provides a comprehensive explanation of secret sharing and presented a few construction methods for secret sharing. Usually, secret sharing requires secure communication channels among the participants to distribute and collect their shares as well as a secure environment in which to reconstruct the secret and compare results. The security of secret reconstruction can also be achieved by other means. Beimel and Chor [4] studied the feasibility of reconstructing the secret over public communication channels and provided complexity bounds for three possible schemes in terms of the ratio between the size of the shares communicating on the public channels and the size of the original secret. Wang and Wong [5] studied the communication efficiency of secret reconstruction and showed that point-to-point secure communication channels for participants during conventional secret reconstruction can be replaced with partial broadcast channels. Vu *et al.* [6] proposed a combined technique to improve the security of secret sharing by choosing only the most reliable servers for storage and encrypting secret shares using passwords.

In some applications it is desirable to perform operations directly on encrypted data instead of plain data. For example, copyright protection can be provided by the buyer-seller watermarking protocol [7], [8], [9], [10], [11], [12] in which an encrypted watermark is embedded into an encrypted image. This allows the content owner and distributer to each independently assert their own rights to the content. Erkin *et al.* [13] proposed a privacy-preserving K-means clustering algorithm to group people in a social network while protecting their privacy-sensitive data by means of computing encrypted distances with homomorphic encryption. Barni *et al.* [14] presented a privacy-preserving classification system of electro cardiogram (ECG) signals using secure evaluation of the linear branching program, where a server can classify the ECG signals without knowledge of the ECG signals and the client is prevented from learning any information about the classification algorithm used by the server. Many common signal processing operations have also been implemented in the encrypted domain such as the Discrete Fourier Transform (DFT) which Bianchi *et al.* [15] implemented in the encrypted domain with radix-2 and the radix-4 fast Fourier algorithms. They also addressed both the merits and limits of composite signal representation and introduced a secure protocol for converting an encrypted signal with composite representation into the encryptions of the individual signal samples [16]. All of these applications make use of an exponentiation operation on encrypted data which is typically computationally intensive.

In this paper we endeavor to design a new secret sharing scheme in the encrypted domain for secure secret reconstruction. First, we introduce the concepts of probabilistic and homomorphic encryption. These cryptographic properties are fundamental to encrypted domain processing. Next a traditional secret sharing scheme is described. We then develop a composite algorithm using binary representation and precomputation for efficient exponentiation of encrypted data. Finally a new scheme of secret sharing in the encrypted domain is proposed based on the efficient exponentiation algorithm.

II. Preliminaries

In this section cryptographic basis and several useful properties for encrypted domain processing are roughly introduced and traditional secret sharing is briefly reviewed.

This work was supported by the endowment of the Charles William Harrison Distinguished Professorship at Purdue University. Address all correspondence to Prof. Edward J. Delp (ace@ecn.purdue.edu).

A. Probabilistic Encryption

Cryptographic techniques are often used to protect sensitive information. If deterministic encryption is employed, two identical plaintext x and y will always result in the same cyphertext. The problem is that an adversary can retrieve information about the plaintext and possibly obtain the original data through a brute force attack. Consequently, probabilistic encryption has to be used for sensitive data [17]. A probablilistic encryption system uses a random number r when generating the ciphertext. In this case, even if x and y are identical, their encrypted representations $E(x)$ and $E(y)$ will differ, where $E(\cdot) = E_{pk}(\cdot, r)$ is the probabilistic encryption function with public key pk and random number r. The plaintext x can be retrieved by decrypting the cyphertext, $x = D(E(x))$, where $D(\cdot) = D_{sk}(\cdot)$ is the decryption function with private key sk. The decryption function does not need knowledge of the random number r used by the encryption function.

B. Homomorphic Encryption

Homomorphism is an important property of some public key cryptosystems which allows addition or multiplication of the plain data by operations on the encrypted data. A public key encryption function $E(\cdot)$ and private key decryption function $D(\cdot)$ are called homomorphic if the composite effect of the two functions is a one to one mapping between an operation $f_1(\cdot)$ in the plain domain and an operation $f_2(\cdot)$ in the encrypted domain [17]. This property is expressed as

$$f_1(x, y) = D(f_2(E(x), E(y))) \tag{1}$$

where the x and y are taken from the plaintext space. Additive and multiplicative homomorphic encryptions are very important for encrypted domain processing.

For an additive homomorphic encryption system, $f_1(\cdot)$ is addition of plain data and $f_2(\cdot)$ is multiplication of encrypted data.

$$x + y = D(E(x) * E(y)) \tag{2}$$

Note that the above equation implies that

$$x * n = D((E(x))^n). \tag{3}$$

Thus, additive homomorphic encryption allows multiplication of x by a known integer n in the plain domain through exponentiation of ciphertext $E(x)$ in the encrypted domain.

For a multiplicative homomorphic encryption system, $f_1(\cdot)$ is multiplication of plain data and $f_2(\cdot)$ is multiplication of encrypted data.

$$x * y = D(E(x) * E(y)) \tag{4}$$

Paillier cryptosystem [18], Damgård-Jurik cryptosystem [19], and Damgård-Geisler-Krøigård cryptosystem [20] are additive homomorphisms while the ElGamal cryptosystem [21] is a multiplicative homomorphism.

C. Encrypted Domain Processing of Negative Numbers

Encryption and decryption in public key cryptosystems are based on number theory of positive integers. Therefore, the input plaintext are required to be positive integers and the output cyphertext are also positive integers. An important consideration is how to deal with negative numbers in the encrypted domain. Special care must be taken to encrypt negative numbers in order to be compatible with the homomorphic property. A possible approach in [22] used a constant additive shift to eliminate negative numbers and correspondingly subtracting this shift after processing. However, this approach only works for addition, not for multiplication. Erkin *et al.* [23] presented a modular inverse approach on ciphertext in which a negative number $-x$ is encrypted as the modular inverse of the ciphertext of the corresponding positive number $(E(x))^{-1}$ based on the additive homomorphic property. It is necessary in this case to make the plaintext space large enough so that overflow due to modular arithmetic is avoided.

D. Traditional Secret Sharing

In Shamir's secret sharing scheme [1], a dealer or a trusted third party (TTP) divides a secret or private data S (e.g. password) into l pieces or secret shares such that:

1. Knowledge of any k or more shares is sufficient to reconstruct the secret S.

2. Knowledge of any $(k-1)$ or fewer shares leaves the secret S completely undetermined.

This scheme is known as (k, l) threshold scheme which is used to share a secret S with l participants, where all l shares are equally likely and the secret S is an element in a finite modular field $\mathbb{Z}_N$. The essential idea of Shamir's (k, l) threshold scheme is that it requires k points to determine a polynomial of degree $(k-1)$.

1) Generating the Shares: The dealer or the TTP constructs a $(k-1)$ degree polynomial by choosing at random $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ in the set of integers $\mathbb{Z}$ and letting the secret S be a_0.

$$y = f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_1 x + a_0. \tag{5}$$

The shares of the secret S are l arbitrarily chosen points $\{(x_i, y_i) : y_i = f(x_i), i \in \{1, \cdots, l\}\}$, and then each participant receives a distinct share in plaintext.

2) Reconstructing the Secret: Given any subset of k distinct shares, the coefficients of the original $(k-1)$ degree polynomial can be determined by Lagrange polynomial interpolation. The secret S is then recovered as the constant term a_0. Lagrange polynomial interpolation is a linear combination of the Lagrange basis polynomials. Given a set of k distinct shares $\{(x_1, y_1), \cdots, (x_k, y_k)\}$, the reconstructed polynomial is

$$y = g(x) = \sum_{i=1}^{k} y_i g_i(x), \tag{6}$$

$$g_i(x) = \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j}. \tag{7}$$

Finally, the reconstructed secret S' can be obtained by

$$S' = g(0) = \sum_{i=1}^{k} y_i g_i(0) = \sum_{i=1}^{k} (y_i * \prod_{j=1, j\neq i}^{k} \frac{0 - x_j}{x_i - x_j}). \quad (8)$$

In traditional secret sharing, this reconstruction process is usually done in a secure environment. The collected shares $\{(x_i, y_i) : y_i = f(x_i), i \in \{1, \cdots, k\}\}$ are disclosed and the reconstructed secret S' is finally revealed to each participant involved.

III. Exponentiation of Encrypted Data

Plaintext x is often represented using 8 to 64 bits. For security, at least 1024 bits per sample in ciphertext $E(x)$ is usually required. Conventional exponentiation based on a series of multiplications is quite time-consuming to compute exponentiation of encrypted data. Gordon [24] provided an extensive survey and summarized many approaches to exponentiation for efficiency improvement.

A. Conventional Algorithm

Based on additive homomorphic property, conventional exponentiation of encrypted data is a series of multiplications of the ciphertext $E(x)$ and it may be adequate for relatively small exponents n.

$$(E(x))^n = \underbrace{E(x) * \cdots * E(x)}_{n} = E(x * n). \quad (9)$$

Equation (9) is an extended series of multiplications of a very large ciphertext $E(x)$ (e.g. at least 1024 bits). For relatively large exponents the computation becomes very time consuming.

B. Composite Algorithm Using Binary Representation and Precomputation

Based on related works [25], [26], [27], a composite algorithm using binary representation and precomputation is developed for efficient exponentiation of encrypted data with a known exponent n. The integer n can be converted to a binary representation with a weighted sum of the values of the digits, where the weights start at 1 for the rightmost position and increase by a factor of 2 for each left position. Inspired by this property, a set of basic exponentiations of $E(x)$ can be precomputed by a chain of squares of the last output, and then a short series of multiplications of the basic exponentiations is delicately performed. Thus exponentiation of encrypted data can be computed more efficiently using the composite algorithm described below.

(Step 1) Obtain the binary representation of the exponent n. Let $M = \lfloor \log_2 n \rfloor$, then $(M + 1)$ is the number of digits.

$$\begin{aligned} n &= (b_M b_{M-1} \cdots b_2 b_1 b_0)_2 = \sum_{m=0}^{M} b_m 2^m \quad (10) \\ &= b_M 2^M + b_{M-1} 2^{M-1} + \cdots + b_1 2^1 + b_0 2^0, \end{aligned}$$

where $b_m \in \{0, 1\}$ is the value of the m-th digit.

TABLE I: Computational Complexity of the Conventional and Composite Algorithms for Exponentiation of Encrypted Data (the Number of Ciphertext Multiplications)

	Conventional Algorithm	Composite Algorithm
Precomputation	0	$0 \leq M \ll n$
Exponentiation	n	$0 \leq U \leq M$
Total	n	$M \leq M + U \leq 2M$

(Step 2) Given the ciphertext $E(x)$, precompute a set of basic exponentiations of $E(x)$, i.e. Ω, by a chain of squares of the last output starting from $E(x)$.

$$\Omega = \{(E(x))^{2^m}, m = 0, 1, \cdots, M - 1, M\}. \quad (11)$$

(Step 3) Obtain the exponentiations of $E(x)$ with exponent n by a short series of multiplications of the basic exponentiations in Ω using the bits in each position as multiplication indicators.

$$\begin{aligned} (E(x))^n &= (E(x))^{\sum_{m=0}^{M} b_m 2^m} = \prod_{m=0}^{M} ((E(x))^{2^m})^{b_m} \\ &= ((E(x))^{2^M})^{b_M} * ((E(x))^{2^{M-1}})^{b_{M-1}} * \quad (12) \\ &\quad \cdots * ((E(x))^{2^1})^{b_1} * ((E(x))^{2^0})^{b_0}. \end{aligned}$$

Compared with the conventional algorithm, the composite algorithm using binary representation and precomputation is more efficient for exponentiation of encrypted data. The number of ciphertext multiplications in the chain of squares for precomputation is M and the one in the short series of multiplications of the basic exponentiations is $U = (\sum_{m=0}^{M} b_m) - 1$. Even though a cost of the precomputation is incurred to generate the set of basic exponentiations, the total number of ciphertext multiplications is significantly reduced from $O(n)$ to $O(M) = O(\lfloor \log_2 n \rfloor)$. Table I shows the computational complexity of the conventional and composite algorithms for exponentiation of encrypted data, in terms of the number of ciphertext multiplications.

IV. Secret Sharing in the Encrypted Domain

A key disadvantage of traditional secret sharing is that the secret and the shares are distributed, stored, and reconstructed in plaintext. The secret and the shares are therefore vulnerable to exposure to adversaries and are subject to many attacks. Another disadvantage is that traditional secret sharing is a one-time operation in the sense that the selected shares and the reconstructed secret cannot be reused any more. To deal with such disadvantages and protect sensitive information, the secret and the shares can be encrypted with the dealer's or the TTP's public key using probabilistic and homomorphic encryption and then the encrypted shares are distributed to each participant. Secret reconstruction could be performed in an insecure environment, where any public and even untrusted devices can be used to process the encrypted inputs and the outputs are also encrypted. In this case, the shares are never revealed to participants and the secret is always maintained secret. The existing encrypted secret and shares are therefore reusable. Therefore, it is desirable to perform operations directly on the encrypted secret and shares for secret sharing.

A. Generating and Encrypting the Shares

The dealer or the TTP constructs a $(k-1)$ degree polynomial similar to Equation (5) by choosing at random $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ in a subset of integers in the finite modular field $\mathbb{Z}_N$ and letting the secret S be a_0. The shares of the secret S are l arbitrarily chosen points $\{(x_i, y_i) : y_i = f(x_i), i \in \{1, \cdots, l\}\}$. For each secret share, y_i is encrypted by the dealer's or the TTP's public key as $E(y_i)$. Each participant receives a distinct share in ciphertext $\{(x_i, E(y_i)) : E(y_i) = E(f(x_i)), i \in \{1, \cdots, l\}\}$.

B. Reconstructing the Secret in the Encrypted Domain

Given any subset of k distinct shares in ciphertext, the coefficients of the original polynomial can be determined by a modified Lagrange polynomial interpolation in the encrypted domain. The secret S is then recovered as the constant term a_0. The modified Lagrange polynomial interpolation is based on the additive homomorphic property. Given a set of k distinct shares $\{(x_1, E(y_1)), \cdots, (x_k, E(y_k))\}$ and scaling factor $c = \prod_{i=1}^{k} \prod_{j=i+1}^{k} (x_i - x_j)$, the modified reconstruction polynomial in the encrypted domain is

$$\begin{aligned} E(cy) &= E(cg(x)) = E(\sum_{i=1}^{k} y_i(cg_i(x))) \qquad (13) \\ &= \prod_{i=1}^{k} E(y_i(cg_i(x))) = \prod_{i=1}^{k} (E(y_i))^{cg_i(x)}, \end{aligned}$$

$$cg_i(x) = \prod_{i=1}^{k} \prod_{j=i+1}^{k} (x_i - x_j) * \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j}, \qquad (14)$$

$$\begin{aligned} E(cS') &= E(cg(0)) = E(\sum_{i=1}^{k} y_i(cg_i(0))) \qquad (15) \\ &= \prod_{i=1}^{k} E(y_i(cg_i(0))) = \prod_{i=1}^{k} (E(y_i))^{cg_i(0)}. \end{aligned}$$

Note that above equation involves exponentiation of encrypted data which can make use of the efficient exponentiation algorithm introduced in Section III-B. Although the same scaling factor c is used for k distinct shares, the base $E(y_i)$ and the exponent $cg_i(0)$ of each exponentiation vary for different shares. Finally, the reconstructed secret S' can be obtained by

$$S' = D(E(cS'))/c = D(\prod_{i=1}^{k} (E(y_i))^{cg_i(0)})/c = g(0) = S. \qquad (16)$$

TABLE II: Average Run Time of Damgård-Jurik Cryptosystem (in Microseconds)

Key	Encryption		Decryption	
Generation	Pos. Int.	Neg. Int.	Pos. Int.	Neg. Int.
106852	35968	37067	70630	71831

TABLE III: Average Run Time of the Conventional and Composite Algorithms for Exponentiation of 2048 Bit Ciphertext (in Microseconds)

Exponent	Conventional Algorithm	Composite Algorithm
$2^4 \sim (2^5-1)$	838 ∼ 1641	232 ∼ 459
$2^8 \sim (2^9-1)$	13312 ∼ 26451	455 ∼ 906
$2^{12} \sim (2^{13}-1)$	214384 ∼ 426701	670 ∼ 1334
$2^{16} \sim (2^{17}-1)$	3390313 ∼ 6790586	887 ∼ 1768
$2^{20} \sim (2^{21}-1)$	54496409 ∼ 108778520	1102 ∼ 2201
$2^{24} \sim (2^{25}-1)$	869783924 ∼ 1750182273	1315 ∼ 2627
$2^{28} \sim (2^{29}-1)$	13971042637 ∼ 28168357444	1532 ∼ 3059
$2^{32} \sim (2^{33}-1)$	225166538223 ∼ 456062147373	1743 ∼ 3483

V. Experiments

Experiments were implemented in Java using the standard BigInteger class of J2SE 1.5.0 [28] and executed on a 3.2 GHz Intel Pentium 4 CPU with Hyper-Threading running Redhat Enterprise Linux 5. Each result was averaged over 500 rounds after an initial warm-up phase of 50 rounds. All times are in microseconds.

For additive homomorphic encryption, Damgård-Jurik cryptosystem was employed with the public key size of 1024 bits (a product of two large primes of 512 bits each). The plaintext space is 1024 bits and the ciphertext space is 2048 bits. Note that Damgård-Jurik cryptosystem with parameter $s = 1$ is equivalent to Paillier cryptosystem [19]. Both positive and negative integers can be used as inputs, where negative integers are mapped as described in [23]. Table II illustrates the average run time of the usage of Damgård-Jurik cryptosystem.

The average run time of the conventional and composite algorithms for exponentiation of 2048 bit ciphertext are shown in Table III. It is clear that the composite algorithm using binary representation and precomputation performs significantly faster than the conventional one. Table IV shows the average run time of the proposed scheme of secret sharing in the encrypted domain. The results demonstrate that it is efficient to be deployed in many practical applications with high security requirement. For example, electronic voting and recommendation system.

VI. Conclusions

This paper proposed a new scheme of secret sharing in the encrypted domain in order to protect each share during secret reconstruction. An efficient algorithm using binary representation and precomputation was developed for exponentiation of encrypted data. The new secret sharing scheme in the encrypted domain made use of this efficient exponentiation algorithm. Experimental results verified the effectiveness of

TABLE IV: Average Run Time of the Proposed Scheme of Secret Sharing in the Encrypted Domain (in Microseconds)

(k, l) Threshold	Generation and Encryption	Reconstruction and Decryption
$(2, 4)$	146139	72406
$(4, 8)$	292685	74886
$(8, 16)$	585962	79743
$(16, 32)$	1183427	98587
$(32, 64)$	2378682	140269

the efficient exponentiation algorithm and the scheme of secret sharing in the encrypted domain. As described in the introduction, many applications can benefit from the efficient exponentiation of encrypted data.

References

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[2] G. Blakley, "Safeguarding cryptographic keys," *Proceedings of the 1979 National Computer Conference*, New York, NY, USA, 4-7 Jun. 1979, pp. 313–317.

[3] D. Stinson, "An explication of secret sharing schemes," *Designs, Codes and Cryptography*, vol. 2, no. 4, pp. 357–390, Dec. 1992.

[4] A. Beimel and B. Chor, "Secret sharing with public reconstruction," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1887–1896, Sep. 1998.

[5] H. Wang and D. Wong, "On secret reconstruction in secret sharing schemes," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 473–480, Jan. 2008.

[6] L.-H. Vu, K. Aberer, S. Buchegger, and A. Datta, "Enabling secure secret sharing in distributed online social networks," *Proceedings of the 25th Annual Computer Security Applications Conference - ACSAC 2009*, Honolulu, HI, USA, 7-11 Dec. 2009, pp. 419–428.

[7] N. Memon and P. Wong, "A buyer-seller watermarking protocol," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 643–649, Apr. 2001.

[8] C.-L. Lei, P.-L. Yu, P.-L. Tsai, and M.-H. Chan, "An efficient and anonymous buyer-seller watermarking protocol," *IEEE Transactions on Image Processing*, vol. 13, no. 12, pp. 1618–1626, Dec. 2004.

[9] M. Kuribayashi and H. Tanaka, "Fingerprinting protocol for images based on additive homomorphic property," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2129–2139, Dec. 2005.

[10] J. Zhang, W. Kou, K. Fan, and L. Ye, "Watermarking protocol of secure verification," *Journal of Electronic Imaging*, vol. 16, no. 4, 043002, pp. 1–4, Oct. 2007.

[11] F. Frattolillo, "Watermarking protocol for web context," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 350–363, Sep. 2007.

[12] B. Zhao, W. Kou, H. Li, L. Dang, and J. Zhang, "Effective watermarking scheme in the encrypted domain for buyer-seller watermarking protocol," *Information Sciences*, vol. 180, no. 23, pp. 4672–4684, Dec. 2010.

[13] Z. Erkin, T. Veugen, T. Toft, and R. Lagendijk, "Privacy-preserving user clustering in a social network," *Proceedings of the 1st IEEE International Workshop on Information Forensics and Security - WIFS 2009*, London, UK, 6-9 Dec. 2009, pp. 96–100.

[14] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Combining signal processing and cryptographic protocol design for efficient ECG classification," *Proceedings of the 1st International Workshop on Signal Processing in the EncryptEd Domain - SPEED 2009*, Lausanne, Switzerland, 10 Sep. 2009, pp. 42–61.

[15] T. Bianchi, A. Piva, and M. Barni, "On the implementation of the discrete fourier transform in the encrypted domain," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 86–97, Mar. 2009.

[16] T. Bianchi, T. Veugen, A. Piva, and M. Barni, "Processing in the encrypted domain using a composite signal representation: pros and cons," *Proceedings of the 1st IEEE International Workshop on Information Forensics and Security - WIFS 2009*, London, UK, 6-9 Dec. 2009, pp. 176–180.

[17] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, 13801, pp. 1–10, 2007.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Proceedings of the 18th International Conference on the Theory and Application of Cryptographic Techniques - EUROCRYPT 1999*, Prague, Czech Republic, 2-6 May 1999, *Lecture Notes in Computer Science*, vol. 1592, pp. 223–238.

[19] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," *Proceedings of the 4th International Conference on Practice and Theory in Public Key Cryptography - PKC 2001*, Cheju Island, South Korea, 13-15 Feb. 2001, *Lecture Notes in Computer Science*, vol. 1992, pp. 119–136.

[20] I. Damgård, M. Geisler, and M. Krøigård, "Efficient and secure comparison for on-line auctions," *Proceedings of the 12th Australasian Conference on Information Security and Privacy - ACISP 2007*, Townsville, Australia, 2-4 Jul. 2007, *Lecture Notes in Computer Science*, vol. 4586, pp. 416–430.

[21] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[22] H. Hacıgümüş, B. Iyer, and S. Mehrotra, "Efficient execution of aggregation queries over encrypted relational databases," *Proceedings of the 9th International Conference on Database Systems for Advanced Applications - DASFAA 2004*, Jeju Island, South Korea, 17-19 Mar. 2004, *Lecture Notes in Computer Science*, vol. 2973, pp. 125–136.

[23] Z. Erkin, A. Piva, S. Katzenbeisser, R. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," *EURASIP Journal on Information Security*, vol. 2007, 78943, pp. 1–20, 2007.

[24] D. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, no. 1, pp. 129–146, Apr. 1998.

[25] P. Scott, S. Simmons, S. Tavares, and L. Peppard, "Architectures for exponentiation in GF(2m)," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 3, pp. 578–585, Apr. 1988.

[26] G. Agnew, R. Mullin, and S. Vanstone, "Fast exponentiation in GF(2n)," *Proceedings of the 7th International Conference on the Theory and Application of Cryptographic Techniques - EUROCRYPT 1988*, Davos, Switzerland, 25-27 May 1988, *Lecture Notes in Computer Science*, vol. 330, pp. 251–255.

[27] E. Brickell, D. Gordon, K. McCurley, and D. Wilson, "Fast exponentiation with precomputation," *Proceedings of the 11th International Conference on the Theory and Application of Cryptographic Techniques - EUROCRYPT 1992*, Balatonfüred, Hungary, 24-28 May 1992, *Lecture Notes in Computer Science*, vol. 658, pp. 200–207.

[28] Biginteger class of j2se 1.5.0. Accessed June 2010. [Online]. Available: http://java.sun.com/j2se/1.5.0/docs/api/java/math/BigInteger.html