Notes on Non-Interactive Secure Comparison in "Image Feature Extraction in the Encrypted Domain with Privacy-Preserving SIFT"

Matthias Schneider
Computer Vision Laboratory
ETH Zurich, Switzerland
schneider@vision.ee.ethz.ch

Thomas Schneider
Engineering Cryptographic Protocols Group
TU Darmstadt, Germany
thomas.schneider@ec-spride.de

ABSTRACT

Protocols for secure comparison are a fundamental building block of many privacy-preserving protocols such as privacy-preserving face recognition or privacy-preserving fingerprint authentication. So far, all existing secure comparison protocols that have been used in practical implementations require interaction.

In recent work, Hsu et al. (IEEE Transactions on Image Processing 2012) propose protocols for privacy-preserving computation of the scale-invariant feature transform (SIFT) in the encrypted domain. Their fundamental building block is a new protocol for performing secure comparisons under additively homomorphic encryption that requires no interaction.

In this paper we present potential for optimization and shortcomings of their secure comparison protocol. More specifically, we show that it 1) allows optimizations by shifting computation from the server to the user, 2) removes the gain that the user has in outsourcing computations to the server, and most importantly is 3) either computationally intractable for the server or insecure. As alternatives we propose to use either interactive comparison protocols or non-interactive somewhat or fully homomorphic encryption.

Categories and Subject Descriptors

F.1.2 [**Modes of computation**]: Interactive and reactive computation—*cryptographic protocols*

Keywords

Signal Processing in the Encrypted Domain; Homomorphic Encryption; Privacy-preserving Comparison

1. INTRODUCTION

Privacy-preserving protocols allow to process sensitive data, signals and multimedia content under encryption, cf. [EPK+07]. A fundamental primitive in many such protocols are protocols for secure comparison that allow two parties to compare their inputs in a privacy-preserving way, e.g., [Yao86, Fis01, BK04, DGK08b, DGK08a, KSS09]. As many privacy-preserving protocols are based on additively homomorphic encryption, secure comparison protocols have been adapted to use ciphertexts as inputs and outputs, e.g., in protocols for privacy-preserving face recognition [EFG+09, SSW09], privacy-preserving fingerprint authentication [BBC+10], or processing encrypted floating point signals [FK11].

All such secure comparison protocols over ciphertexts known so far require interaction between the parties. An intuitive reason for this is that additively homomorphic encryption allows only to perform linear operations (i.e., addition or multiplication by a constant) under encryption, whereas comparison is an inherently non-linear operation. Hence, secure comparison of encrypted values is often considered to be an expensive operation, see e.g., [EPK+07, EBVL12].

One possibility to avoid interaction is to use fully homomorphic encryption schemes that allow both addition and multiplication under encryption. Such schemes were recently introduced by Gentry [Gen09a, Gen09b] and many optimizations and alternative schemes have been proposed, e.g., [DGHV10, SS11, BV11b, BV11a, CNT12, BGV12, GHS12a, GHS12b]. Although first implementations of fully homomorphic encryption have emerged, e.g., [SV10, GH11, LNV11, GHS12c], these implementations are currently not efficient enough to be used in larger privacy-preserving applications.

Recently, a system for privacy-preserving computation of the scale-invariant feature transform (SIFT) in the encrypted domain has been proposed in [HLP12]. Their fundamental building block is a new protocol for performing secure comparisons under *additively homomorphic encryption* that requires *no interaction*.

Our Contributions. In §3 of this paper we present potential for optimization and shortcomings of [HLP12] which are also present in earlier versions of that article [HLP10, HLP11]. More precisely, the non-interactive protocol for secure comparison of additively homomorphically encrypted values proposed in these works 1) allows optimizations by shifting computation from the server to the user, 2) removes the gain that the user has in outsourcing computations to the server, and most importantly is 3) either computationally intractable for the server or insecure. In §4, we summarize alternative solutions from the literature.

Outline. In §2 we give necessary preliminaries and summarize the comparison protocol of [HLP12] and its application for privacy-preserving SIFT. As our contributions we present potential for optimization and shortcomings of the protocol in §3 and give alternative solutions from the literature in §4. We conclude in §5.

IH&MMSec'14, June 11–13, 2014, Salzburg, Austria.

http://dx.doi.org/10.1145/2600918.2600927.

2. BACKGROUND ON HSU ET AL.'S COMPARISON PROTOCOL

We first summarize the additively homomorphic encryption scheme of Paillier in §2.1, the protocol for privacy-preserving SIFT of [HLP12] in §2.2, and their non-interactive secure comparison protocol in §2.3.

2.1 Additively Homomorphic Encryption

The protocols of [HLP12] are based on the additively homomorphic cryptosystem of Paillier [Pai99] as described next.

The encryption of a message $m \in Z_N^*$ using randomness $r \in_R Z_N^*$ is computed as $E(m,r) = g^m r^N \bmod N^2$, where the public key consists of $g \in Z_{N^2}^*$ and an RSA modulus $N = pq$, where p and q are large primes. Early versions of the paper [HLP10,HLP11] recommended to use 100 bit primes p, q which is too small to provide security in practice. The journal version [HLP12, Sect. VI] proposes to use 1 000 bit primes which is in accordance with the 1 024 bits proposed in current recommendations for key lengths[1].

The cryptosystem is additively homomorphic, i.e., it allows to add two messages under encryption:

$$\begin{aligned} E(m_1,r_1) \cdot E(m_2,r_2) \bmod N^2 &= g^{m_1+m_2}(r_1 r_2)^N \bmod N^2 \\ &= E(m_1 + m_2 \bmod N, r_1 r_2 \bmod N^2). \end{aligned}$$

(**Note** that for addition of messages the randomness is multiplied.)

It is also possible to multiply an encrypted message with a constant a:

$$\begin{aligned} E(m,r)^a \bmod N^2 &= g^{am} r^{aN} \bmod N^2 \\ &= E(am \bmod N, r^a \bmod N^2). \end{aligned}$$

(**Note** that for multiplying the message with a the randomness gets raised to the a-th power.)

Informally speaking, the semantic security of the Paillier cryptosystem implies that it is not possible to infer any information about the encrypted plaintext from seeing the ciphertext and the public key only. In particular, it is not possible to compare two ciphertexts when knowing only the public key.

The cryptosystem can also be extended to have a larger plaintext-space and optimized such that encryption costs about one modular exponentiation [DJ01].

2.2 Privacy-Preserving SIFT

The authors of [HLP12] propose to use the Paillier cryptosystem for privacy-preserving SIFT. Here, the user wants to outsource the computation of the SIFT algorithm on an image to the server in such a way that the server does not learn any information about the image. In the first step, the server applies the Difference-of-Gaussian (DoG) transform in the encrypted domain: First, he receives from the client the pixel-wise encrypted image $I_e(x,y) = E(I(x,y), r_{x,y})$ for all pixels (x,y) of the image $I(x,y)$, where $r_{x,y}$ is a randomly chosen value used for encryption. Afterwards, the DoG filter $G_{\mathrm{Diff}}(u,v,\rho = (\rho_i, \rho_j))$, defined as the scaled difference of two Gaussian kernels at scales ρ_i and ρ_j with DoG filter coefficients eventually rounded to integer values (cf. Eq. (9) in [HLP12]), is applied to the encrypted image by computing

$$\begin{aligned} \mathrm{DoGImg}_e(x,y,\rho) &= \prod_{u,v} I_e(x-u, y-v)^{G_{\mathrm{Diff}}(u,v,\rho)} \bmod N^2 \\ &= E(\mathrm{DoGImg}(x,y,\rho), R_\rho) \end{aligned}$$

(cf. Eq. (11) and Eq. (12) in [HLP12]).

[1] `http://keylength.com`

The resulting randomness is $R_\rho = \prod_{u,v} r_{x-u,y-v}^{G_{\mathrm{Diff}}(u,v,\rho)}$ (cf. Eq. (13) in [HLP12]). Afterwards, the server should detect local extrema of these transformed images in the encrypted domain. For this, the authors of [HLP12] propose a secure comparison protocol described next.

2.3 The Comparison Protocol of Hsu et al.

To allow the server to compare two ciphertexts $E(m_1,r_1)$ and $E(m_2,r_2)$, the authors of [HLP12] propose the following protocol: First, the server reveals to the user the random values r_1 and r_2. As the server does not know these random values, he reveals to the user the sequence of operations that he has applied to the initial ciphertexts obtained by the user. This allows the user to compute the random values (cf. notes in §2.1 above). Afterwards, the user chooses an increasing sequence of random thresholds $T_i \in Z_N$ and encrypts them using the same random values r_1 and r_2. He sends $E(T_i,r_1)$ and $E(T_i,r_2)$ to the server. Now, the server computes the distance a_{k_1} between the first encrypted message and the closest encrypted threshold with index t_{k_1} as

$$(a_{k_1}, t_{k_1}) = \arg\min_{\mathrm{Inc},i} \big(E(m_1,r_1) g^{\mathrm{Inc}} - E(T_i,r_1)\big). \quad (1)$$

The authors of [HLP12] propose to compute this by repeatedly multiplying $E(m_1,r_1)$ with g until after $a_{k_1} = \mathrm{Inc}$ times this value is equal to the encrypted threshold with index $t_{k_1} = i$. After having computed a_{k_2} and t_{k_2} in a similar way, it is possible to determine whether $m_1 < m_2$ or not. To speed up computations, the server (or the user) could also pre-compute a lookup table, but this essentially shifts the computations from the online phase into a setup phase.

3. NOTES ON HSU ET AL.'S COMPARISON PROTOCOL

The comparison protocol of [HLP12] has potential for optimization and shortcomings as described next.

3.1 Potential for Optimization and Alternatives

To allow the user to compute the random values r_1 and r_2, the server needs to reveal to the user the exact sequence of operations and parameters that he has applied to the ciphertexts. For example, in the setting of privacy-preserving SIFT, the server reveals the Gaussian coefficients $G_{\mathrm{Diff}}(x,y,\rho)$ (cf. [HLP12, Fig. 3]). Hence, the user knows all operations that the server has performed before the comparison and hence can apply the operations himself to the plaintexts before sending the encrypted values to the server. This allows to shift computation from the server to the user, but can increase the amount of data sent from the user to the server. As performing operations on plaintexts is substantially faster than on ciphertexts, this is a viable solution for many applications, cf. [Ker11]. Wagner et al. [WRM+10] have even shown that variations of SIFT features can efficiently be computed in real time on mobile devices facing limited computational resources. In this case, the encrypted feature descriptors rather than the image data would have to be transmitted from the client to the server. Computing the SIFT descriptors in the plaintext domain would also allow to overcome most of the simplifications of PPSIFT over SIFT, e.g., rounding of Gaussian coefficients [HLP12, Eq. (9)], four restrictive gradient directions [HLP12, Sect. IV.C], no accurate keypoint localization [Low04, Sect. 4]. Furthermore, shifting the feature computation from the server to the client side even allows to approach the problem of secure image retrieval without the Paillier-based privacy-preserving SIFT evaluation of [HLP12]. Instead, a visual words representation of the query image based on SIFT features can

be used along with more efficient cryptographic techniques such as random permutations or order preserving encryption as proposed in [LVSW09, LSVW09]. For this, additional computational effort is required on the client side as the SIFT features of the query image have to be quantized with respect to a codebook first. The codebook can be computed, e.g., by hierarchically clustering the SIFT descriptors of the remote server database into a vocabulary tree. To the best of our knowledge, a detailed performance comparison between the techniques proposed in [LVSW09, LSVW09] and the protocol of [HLP12] has not been published yet (we will show later in §3.3 that the protocol of [HLP12] is either completely impractical or insecure).

3.2 High Computational Effort for the User

Now, as the user knows the operations that the server wants to apply under encryption, he could perform these operations by himself as well: he performs the computations in the plaintext domain and afterwards generates a fresh encryption of the result (which costs about one modular exponentiation [DJ01]), i.e., instead of re-computing the same random value for the encrypted thresholds (cf. §2.3), he chooses a new random value for the encryption and the encrypted thresholds. For example, in the setting of privacy-preserving SIFT, the user could use the Gaussian coefficients to apply the Gaussian filter himself and afterwards generate a fresh encryption of the filtered image which is sent to the server. In fact, the computational effort for applying the operation on the plaintexts is smaller than computing the random values (cf. notes in §2.1 above): adding two plaintexts requires one modular addition (instead of a modular multiplication for computing the randomness) and multiplying a plaintext with a constant requires one modular multiplication (instead of a substantially more expensive modular exponentiation). Hence, in the proposed comparison protocol, the client has no advantage in outsourcing all computations done before the comparison to the server any more.

3.3 Infeasible Computational Effort for the Server or Insecurity

Next we show that the protocol is either computationally infeasible or insecure, depending on the size of the encrypted values.

Infeasible computational effort for large values. For sufficiently large encrypted values, we show that the amount of computation that the server needs to perform to evaluate Eq. (1) is not feasible when the security parameters are chosen according to today's recommendations. Assuming that 10 thresholds are chosen at random and the primes p, q have 1000 bits (as recommended in [HLP12, Sect. VI]). Then, the plaintext space Z_N has about 2^{2000} elements and the distance to the next threshold is on average $2^{2000}/(2 \cdot 10) > 2^{1995}$. Today's fastest supercomputer, the IBM Sequoia, has a performance of 16.32 petaflops, i.e., it can perform $16.32 \cdot 10^{15}$ floating point operations per second. Even when assuming that one step in the computation of Eq. (1) could be performed at the time of a single floating point operation, this would require more than $2^{1995}/(16.32 \cdot 10^{15}) > 2^{1941}$ seconds which is far beyond the lifetime of our universe. Note that using more thresholds would also increase the communication in the setup phase and essentially shifts computations from the server to the user: Even with 2^{80} thresholds, which results in 1000 YottaBytes ($= 1000 \cdot 10^{24}$ Bytes) of initial communication (this is more than the total amount of information stored on the Internet today), the distance to the next threshold would still be more than 2^{1919} operations which remains computationally infeasible.

Insecurity for small values. We have shown above that the comparison protocol is computationally infeasible when the encrypted values are large, i.e., taken from the entire plaintext space. When the encrypted values are taken from a smaller domain, the complexity of the protocol can be reduced, but this completely breaks security as we show next. Indeed, in the usage scenario for the SIFT application the image space, i.e., the range for the representation of pixel values, is between 0 and 255 in [HLP12], which is much smaller than the plaintext space of the cryptosystem. (There is some rescaling applied for the Gaussian filter, but the image space still remains much smaller than the plaintext space. More concretely, the authors of [HLP12] propose to use a scaling factor $s = 2^{24}$ such that the largest value would be at most $255 \cdot 2^{24} < 2^{32}$ which is relatively small.) Now, the thresholds can also be taken from the same subspace s.t. the entire range in between thresholds will also be just a subset of the ciphertext space (because the same randomness is used for encrypting the thresholds). This smaller image space makes the comparison protocol computationally feasible, but completely insecure. As the same encrypted thresholds are used for comparison, it is in fact possible for a curious server to completely break the security of the proposed comparison protocol as follows: First, the server computes the distance between the two encrypted messages m_1 and m_2 to their next threshold using Eq. (1) twice. Afterwards, he substitutes in this equation $E(m_1, r_1)$ and $E(T_i, r_1)$ with the two encrypted thresholds to compute their distance. Adding these three distances yields the distance between m_1 and m_2. This information is not to be revealed in secure protocols for comparison which are described below. Hence, substantially improving the computational complexity of the protocol while attaining security is an important open problem, cf. [HLP12, Sect. VII].

Another observation made as early as in [RAD78] is that any homomorphic cryptosystem that allows non-interactive comparisons of ciphertexts and reveals the result of this comparison in the clear is insecure, as it allows the adversary to decrypt a given ciphertext using a binary search strategy.

4. ALTERNATIVE SECURE COMPARISON PROTOCOLS

As shown in §3, the non-interactive solution based on additively homomorphic encryption proposed in [HLP12] has several shortcomings and hence cannot be used in practice (unless short key lengths are used [HLP10, HLP11] which undermine security). Currently, there is no solution available that is both 1) non-interactive and 2) requires only additively homomorphic encryption. The only solution known so far is to drop one of the requirements, i.e., either drop 1) and keep 2) by using interactive protocols (§4.1), or keep 1) and drop 2) by using more powerful but slower somewhat or fully homomorphic encryption (§4.2).

4.1 Interactive Protocols using Additively Homomorphic Encryption

Existing and provably secure protocols for secure comparisons can be used instead, e.g., [BK04, DGK08b, DGK08a, KSS09]. Such protocols have been adapted to compare values encrypted with an additively homomorphic encryption scheme, e.g., in protocols for privacy-preserving face recognition [EFG+09, SSW09] or privacy-preserving fingerprint authentication [BBC+10]. However, these secure comparison protocols would require interaction between the user and the server which should be avoided in the framework of [HLP12], cf. footnote on page 3 of their paper. We propose a non-interactive solution next.

Another approach that minimizes the interaction between the user and the server is to outsource computations not only to a single server but to two (or more) non-colluding servers among which an interactive secure computation protocol is run. This approach was taken in many applications, e.g., [FPRS04, BLW08, BCD+09], and can result in very efficient solutions.

4.2 Non-Interactive Comparison using Somewhat or Fully Homomorphic Encryption

For a non-interactive solution, fully-homomorphic encryption schemes could be used as described in §1. Boolean circuits for secure comparison can be built with logarithmic multiplicative-depth[2], e.g., the circuit described in [GSV07] has multiplicative depth $\lceil \log_2 \ell \rceil$ for comparing ℓ-bit values. Hence, it is also possible to use somewhat homomorphic encryption schemes, e.g., [Gen09b, DGHV10,BV11b,BV11a], that allow a fixed number of multiplications of ciphertexts. In contrast to fully-homomorphic encryption schemes, these schemes do not require an expensive bootstrapping step and hence can be implemented more efficiently, cf. [LNV11]. The authors of [LNV11] report on practical implementation results for the somewhat homomorphic encryption scheme of [BV11b] where parameters are chosen to allow up to 15 multiplications, i.e., it can be used to non-interactively compare numbers of of up to $2^{15} = 32\,768$ bits which is sufficient for the privacy-preserving SIFT application of [HLP12] where numbers fit into the plaintext space with $|N| = |p| + |q| = 2\,000$ bits. The performance reported in [LNV11, Tab. 2] is in the order of few seconds per operation which makes non-interactive secure comparison feasible. However, this solution requires that the inputs are given as encrypted bits. To decompose encrypted integers into encrypted bits can be done using an interactive bit decomposition protocol, e.g., [DFK+06, ST06], or using fully homomorphic encryption.

5. CONCLUSION

In this paper we showed potential for optimization and shortcomings of the non-interactive comparison protocol using additively homomorphic encryption of [HLP12]. One solution to get a practical protocol is to use short keys, as proposed in earlier versions of the paper [HLP10, HLP11], but this does not provide enough security. For keys of reasonable size the protocol gets only computationally feasible when the encrypted values and thresholds are from a small domain, but then the protocol is insecure. In order to get a secure *and* computationally feasible solution that can be implemented in a real system, we propose to drop one of the requirements and use either interactive comparison protocols or more powerful, but still computationally feasible somewhat or fully homomorphic encryption. Finding a solution that is secure, computationally feasible, non-interactive, and uses only *additively* homomorphic encryption remains an open research problem. An interesting direction for future work might be to consider also secure computation of other descriptors, e.g., BRIEF [CLSF10] or SURF [BTG06], and compare their performance with secure computation of SIFT.

Acknowledgments

We thank the anonymous reviewers of IH&MMSec'14 for their helpful comments. The second author was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, by the Hessian LOEWE excellence initiative within CASED, and by the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement n. 609611 (PRACTICE).

[2]The multiplicative depth of a circuit is its maximum number of multiplications (AND gates) on any path from an input to an output.

6. REFERENCES

[BBC+10] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Fabio Scotti, and Alessandro Piva. Privacy-preserving fingercode authentication. In *ACM Workshop on Multimedia and Security (MM&Sec'10)*, pages 231–240. ACM, 2010.

[BCD+09] Peter Bogetoft, Dan L. Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus D. Nielsen, Jesper B. Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security (FC'09)*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS'12)*, pages 309–325. ACM, 2012.

[BK04] Ian F. Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Advances in Cryptology – ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 515–529. Springer, 2004.

[BLW08] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security (ESORICS)*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc Gool. SURF: Speeded up robust features. In *European Conference on Compute Vision (ECCV'06)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Foundations of Computer Science (FOCS'11)*, pages 97–106. IEEE, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology – CRYPTO'11*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary robust independent elementary features. In *European Conference on Compute Vision (ECCV'10)*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer, 2010.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

[DFK+06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party

computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2006.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[DGK08a] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. A correction to "efficient and secure comparison for on-line auctions". Cryptology ePrint Archive, Report 2008/321, 2008.

[DGK08b] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Homomorphic encryption and secure comparison. *Journal of Applied Cryptology*, 1(1):22–31, 2008.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public-Key Cryptography (PKC'01)*, Lecture Notes in Computer Science, pages 119–136. Springer, 2001.

[EBVL12] Zekeriya Erkin, Michael Beye, Thijs Veugen, and Reginald L. Lagendijk. Privacy-preserving content-based recommender system. In *ACM Workshop on Multimedia and Security (MM&Sec'12)*, pages 77–84. ACM, 2012.

[EFG+09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies (PET'09)*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.

[EPK+07] Zekeriya Erkin, Alessandro Piva, Stefan Katzenbeisser, Reginald L. Lagendijk, Jamshid Shokrollahi, Gregory Neven, and Mauro Barni. Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. *EURASIP Journal on Information Security*, 2007.

[Fis01] Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Cryptographer's Track at RSA Conference (CT-RSA'01)*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–472. Springer, 2001.

[FK11] Martin Franz and Stefan Katzenbeisser. Processing encrypted floating point signals. In *ACM Workshop on Multimedia and Security (MM&Sec'11)*, pages 103–108. ACM, 2011.

[FPRS04] Joan Feigenbaum, Benny Pinkas, Raphael Ryger, and Felipe Saint-Jean. Secure computation of surveys. In *EU Workshop on Secure Multiparty Protocols (SMP)*. ECRYPT, 2004.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Symposium on Theory of Computing (STOC'09)*, pages 169–178. ACM, 2009.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology – EUROCRYPT'11*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography (PKC'12)*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology – EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology – CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[GSV07] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography (PKC'07)*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2007.

[HLP10] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. Homomorphic encryption-based secure SIFT for privacy-preserving feature extraction, July 12, 2010. Technical Report No. TR-IIS-10-006.

[HLP11] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. Homomorphic encryption-based secure SIFT for privacy-preserving feature extraction. In *IS&T/SPIE Media Watermarking, Forensics, and Security*, volume 7880, pages 788005–1–788005–17, 2011.

[HLP12] Chao-Yung Hsu, Chun-Shien Lu, and Sao-Chang Pei. Image feature extraction in encrypted domain with privacy-preserving SIFT. *IEEE Transactions on Image Processing*, 21(11):4593–4607, November 2012.

[Ker11] Florian Kerschbaum. Automatically optimizing secure computation. In *Computer and Communications Security (CCS'11)*, pages 703–714. ACM, 2011.

[KSS09] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology And Network Security (CANS'09)*, volume 5888 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2009.

[LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *ACM Cloud Computing Security Workshop (CCSW'11)*, pages 113–124. ACM, 2011.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSVW09] Wenjun Lu, Ashwin Swaminathan, Avinash L. Varna, and Min Wu. Enabling search over encrypted multimedia databases. In *Media Forensics and Security*, volume 7254 of *SPIE Proceedings*, page 725418. SPIE, 2009.

[LVSW09] Wenjun Lu, Avinash L. Varna, Ashwin Swaminathan, and Min Wu. Secure image retrieval through feature protection. In *IEEE International*

Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 1533–1536. IEEE, 2009.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[RAD78] Ronald Rivest, Len Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

[SS11] Peter Scholl and Nigel P. Smart. Improved key generation for Gentry's fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 10–22. Springer, 2011.

[SSW09] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology (ICISC'09)*, volume 5984 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2009.

[ST06] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for paillier encrypted values. In *Advances in Cryptology – EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 522–537. Springer, 2006.

[SV10] Nigel P. Smart and Fre Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography (PKC'10)*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[WRM+10] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, 2010.

[Yao86] Andrew C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS'86)*, pages 162–167. IEEE, 1986.