UNIVERSITÄT SALZBURG

Erratum: "Customisation of Paillier Homomorphic Encryption for Efficient Binary Biometric Feature Vector Matching" does NOT enable Privacy-Preserving Matching

Andreas Uhl

Technical Report 2014-06 October 2014

Department of Computer Sciences

Jakob-Haringer-Straße 2
5020 Salzburg
Austria
`www.cosy.sbg.ac.at`

Technical Report Series

Erratum: “Customisation of Paillier Homomorphic Encryption for Efficient Binary Biometric Feature Vector Matching” does NOT enable Privacy-Preserving Matching

Andreas Uhl
University of Salzburg, Department of Computer Sciences
J.-Haringerstr.2, 5020 Salzburg, Austria
andreas.uhl@sbg.ac.at

Abstract: A privacy-preserving biometric matching technique for binary vectors more efficient than the Goldwasser-Micali approach has been proposed in “Customisation of Paillier Homomorphic Encryption for Efficient Binary Biometric Feature Vector Matching [PPRU14]” based on exploiting Paillier’s capability of encrypting messages larger than one bit at a time. We demonstrate that the suggested solution does not allow privacy preserving matching and show how at least the efficient encryption and XOR-computation part can be used in privacy preserving manner.

1 Introduction

To compare two binary biometric templates m_1 and m_2 we usually calculate the Hamming distance h of the two binary strings by xor-ing $m' = m_1 \oplus m_2$ and then computing the Hamming Weight HW of the resulting string (by essentially counting the 1 bits in m'): $h = \mathcal{HW}(m')$.

Hence, in order to compute h in a privacy preserving way, one option is to find a way of xor-ing two bit-strings in the encrypted domain. For the Goldwasser-Micali encryption scheme we can directly exploit its homomorphic property [GM82], accepting a significant computational cost [PPRU14]. For the Paillier cryptosystem however [Pai99], calculating the xor of two encrypted binary strings is not as trivial. Thus we will have to look at the process of xor-ing two bit-strings more closely.

Let $m_1 = (m_1[1] \dots m_1[k]), m_2 = (m_2[1] \dots m_2[k])$ be two binary strings of length k. Then

$$m_1 \oplus m_2 = m_1[i] + m_2[i] - 2m_1[i]m_2[i], i = 1, \dots, k\,.$$

As a consequence, we have to use bit-by-bit encryption of the Paillier scheme and can finally perform the encrypted xor as follows:

$$\tilde{m_2}[i] = -2m_2[i] \bmod n$$

$$\mathcal{E}_P(m_1[i] \oplus m_2[i]) = \mathcal{E}_P(m_1[i]) \cdot \mathcal{E}_P(m_2[i]) \cdot (\mathcal{E}_P(m_1[i]))^{\tilde{m_2}[i]} \bmod n^2$$

where n is part of the public key for the Paillier cryptosystem and all encryption steps also use the public key.

However, encrypting only a single bit is very inefficient as the Paillier scheme is designed to encrypt messages of length $m < n$ [PPRU14]. To improve performance, in [PPRU14] we proposed a different algorithm for xor-ing two binary m_1, m_2 in the encrypted domain which exploits Paillier's property to encrypt messages of length $m < n$. This algorithm has been termed *Paillier Chunkwise* as the $\oplus$ operation can be applied to chunks of the binary feature vectors in the form of intergers instead of the inefficient bitwise application. The introduced technique fundamentally relies on the subsequent corollary [PPRU14].

Corollary 1 *Before encrypting m_1 and m_2, these messages are up-sampled as follows:* $m_{j,up}[i] = \begin{cases} m_j[\frac{i}{2}], & 2 \mid i \\ 0, & otherwise \end{cases} \quad i = 1 \dots 2 \cdot length(m),\ j \in \{1, 2\}.$
Then the result of xoring m_1 and m_2 can be computed as (for $i = 1 \dots length(m)$):

$$m_1[i] \oplus m_2[i] = (\mathcal{D}_P(\mathcal{E}_P(m_{1,up}) \cdot \mathcal{E}_P(m_{2,up}) \bmod n^2))[2i] \ .$$

This means that when upsampling the binary data of the m_j before encryption, every other binary position of the result of the multiplication of the decimal numbers corresponding to upsampled m_j are equal to the encrypted result of xor-ing the m_j's binary positions. So far, everything's fine, and we end up with an efficient and privacy preserving computation of the xor of m_1 and m_2.

2 Privacy-preseration of Matching is Flawed during Hamming Weight Computation

Regardless of the underlying xor-algorithm, the calculation of HW cannot be accomplished in the encrypted domain. Thus, according to the security architecture outlined in [PPRU14], the result of the xor computation above is transfered to the matcher M, decrypted using the private key and the bits at the even positions (corresponding to the xor-result due to the corollary) are extracted and summed up to obtain HW.

However, there arises a problem with the bits at the uneven positions, which can be equally recovered by M after decryption. According to the proof of Lemma 1 given in [PPRU14], these positions contain the carry bits the the "upsampled" addition operation. Unfortunately, these carry bits can be used to partially recover the original bitstrings m_1 and m_2, thus violating the desired property of privacy-preservation.

Corollary 2 *From the two bits $m_1[i]$ and $m_2[i]$, on average 1.5 bits may be recovered when analysing the decrypted result vector $res = (\mathcal{D}_P(\mathcal{E}_P(m_{1,up}) \cdot \mathcal{E}_P(m_{2,up}) \bmod n^2))$ at positions [2i] and [2i+1], assuming a uniform distribution of the original feature bits.*

Proof: Considering the decrypted result vector res we may distinguish three cases:

1. $res[2i] = 0 \wedge res[2i+1] = 0 \implies m_1[i] = 0 \wedge m_2[i] = 0.$
2. $res[2i] = 0 \wedge res[2i+1] = 1 \implies m_1[i] = 1 \wedge m_2[i] = 1.$
3. $res[2i] = 1 \wedge res[2i+1] = 0 \implies m_1[i] = 1 \wedge m_2[i] = 0$ or $m_1[i] = 0 \wedge m_2[i] = 1.$

Thus, in case of a zero at position [2i] we may entirely recover $m_1[i]$ and $m_2[i]$, in case of a 1 at this position we know at least that $m_1[i]$ and $m_2[i]$ must have been different. **QED**.

3 Regaining Privacy-preserving Matching employing Secure Multiparty Computations

One approach to avoid the generation of res on the matcher M thus partially revealing $m_1[i]$ and $m_2[i]$ is to only pass the encrypted bits of $(\mathcal{E}_P(m_{1,up}) \cdot \mathcal{E}_P(m_{2,up}) \bmod n^2)$ at positions $[2i]$ to the matcher. To these bits, a bitwise Paillier decryption is applied subsequently. In this manner, HW can be computed in privacy preserving manner, however, at a significantly increased computational cost as shown in Fig. 3(c) of [PPRU14], even more costly as compared to Goldwasser-Micali decryption.

In this context, the BITREP gate technique [ST06], relying on secure multiparty computations, may be employed to privately extract the encrypted bits. Of course, this does not come for free: The computational cost for the matching stage (as shown in Fig.3(b) [PPRU14]) increases substantially and will hardly stay below the cost for Goldwasser-Micali.

4 Conclusions

In recent work [PPRU14] we have proposed a variant of Paillier homomorphic encryption clearly outperforming the Goldwasser-Micali approach in privacy-preserving biometric matching of binary templates. While obtaining significant performance improvements, it has been observed that this approach does in fact not allow for privacy preserving matching. When applying secure multiparty computation to avoid the observed data leakage, we are able to maintain the highly efficient encryption and privacy preserving xor-computation (see Fig.3(a) [PPRU14], about 8-times faster as compared to Goldwasser-Micali), while the performance of the matching and decryption stage gets clearly lower as comapred to the Goldwasser-Micali approach. Thus, for application environments where encryption and xor-computations has to be fast or is executed on weak hardware platforms, the proposed solution might still be of relevance.

5 Acknowledgements

We would like to thank Julien Bringer and Stefan Katzenbeisser for exposing the weakness of the scheme in a private and friendly manner. Furthermore, we thank Stefan Katzenbeisser for providing references for extracting encrypted bits by secure multiparty computation, the application of which allows to at least preserve the benefits of the proposed scheme for the encryption and xor-computation phase.

References

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[PPRU14] Georg Penn, Gerhard Pötzelsberger, Martin Rohde, and Andreas Uhl. Customisation of Paillier Homomorphic Encryption for Efficient Binary Biometric Feature Vector Matching. In *Proceedings of the International Conference of the Biometrics Special Interest Group (BIOSIG'14)*, Darmstadt, Germany, September 2014.

[ST06] B. Schoenmakers and P. Tuyls. Efficient binary conversion for Paillier encrypted values. In *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes on Computer Science, page 522â537, Berlin, Heidelberg, 2006. Springer-Verlag.