Evidence Acquisition in Cloud Forensics

Mahmoud Nasreldin
Ain Shams University
Cairo, Egypt
mahmoudnasreldin@gmail.com

Heba Aslan
ERI, Ministry of Higher Education and Scientific Research, Cairo, Egypt
hebaaslan@eri.sci.eg

Mohamed Rasslan*
ERI, Ministry of Higher Education and Scientific Research, Cairo, Egypt
mohamed@eri.sci.eg

George Weir
Computer and Info. Sciences, University of Strathclyde
Glasgow, United Kingdom
george.weir@strath.ac.uk

Abstract *- In this paper, we present a performance comparison between different digital evidence acquisition protocols in the cloud-computing environment. We focus on data confidentiality, authenticity, and integrity issues.*

Keywords— C*loud Computing; Digital Forensics; RSA; Elliptic Curve Cryptography*

I. INTRODUCTION

To protect the privacy of uninvolved users and prevent collateral intrusion in criminal investigations, one approach is to let the cloud's administrator send the relevant data to the investigator and keep other users' data secure. On the other hand, the investigator might want to keep the administrator away from the investigation process. In this case, it is crucial to protect both the confidentiality and privacy of the investigation. To solve this problem, Hou *et al.* [1, 2] presented several solutions under the assumption that the cloud administrator is willing to cooperate with the investigator when searching for the relevant data. In this solution, the administrator is supposed to have responsibility for protecting the innocuous data from disclosure, yet at the same time, not allowed to know about the relevant data. However, without learning what the relevant data is, the administrator cannot judge if the relevant data is truly relevant to the crimes and cannot verify the investigator obtained other irrelevant data from the server. With respect to this problem, Hou *et al.* assume that the administrator can require the investigator to show what data was collected based on what keyword(s) were used when the relevant data is presented as evidence in court [1,2]. However, even if the assumption works, no measures can guarantee that the presented data came from the server and that no alteration had taken place. In other words, authenticity and integrity of the evidence collected in the work [1, 2] are not considered. The authenticity and integrity are two fundamental requirements for admissibility of evidence in court and they are crucial to winning a case. Therefore, we set forth our suggestions on how to prove the authenticity and integrity of the type of evidence collected in the Hou *et al.*' work [1, 2].

In [3], Hou *et al.* proposed an "encryption-then-blind signature with designated verifier" scheme to prove the authenticity and integrity of the evidence in cloud environment. Hou *et al.* aim to improve the investigation efficiency and protect the privacy of innocent user. One strategy is to allow the server administrator to search, retrieve and hand only the relevant data to the investigator, where the administrator is supposed to be responsible for managing the data in a secure manner. There may be some elements due to their sensitivity or confidential nature that is might be prudent not to alert the administrator about it. In short, it is indispensable to consider how to protect both the confidentiality of the investigation and the privacy of uninvolved users in such forensic investigations. For simplicity of description, Hou *et al.* refer to this problem as "server-aided confidential forensic investigation". When the above-mentioned relevant data is presented as evidence during a trial, Hou *et al.* try to ensure that the administrator (or a third party that the administrator trusts) can verify whether the presented evidence is data that comes from the server and whether the evidence was altered or not. To mitigate these security breaches, Sign-Encrypt-Sign and Encrypt-Sign-Encrypt techniques are used [4]. Sign-Encrypt-Sign and Encrypt-Sign-Encrypt suffers from computation, implementation, and communication overheads. Zheng [5] introduced the term signcryption in order to achieve greater efficiency than performing the signature and encryption operations separately.

Zawoad *et al.*'s research [6-10], concludes that reliability and accuracy of evidence are very important factors when evaluating evidence during a criminal investigation and prosecution. Thus, they offered to preserve the integrity of evidence before and after collecting it from the cloud environment by first identifying the required properties to support trustworthy forensics in the cloud. Based on the requirements, they proposed a forensics-enabled cloud architecture (FECloud) to preserve and provide required evidence while protecting the privacy and integrity of the evidence. FECloud is designed on top of Open stack that is a popular open source cloud-computing platform. Incorporating architectures like FECloud may impose significant business impacts on Cloud Service Providers (CSP), as well as, customers. CSPs can attract more customers with the assurance of providing proper forensics support. Likewise, customers do not require large investment to establish their

*corresponding author

own forensics-friendly infrastructures. Based on the requirements, Zawoad *et al.* proposed a forensics-friendly architecture, FECloud. FECloud introduces five new components in the existing OpenStack architecture namely: Logger (Themis), Data Possession (DP) Manager (Metis), Timestamp Manager (Chronos), Provenance Manager (Clio), and Proof Publisher (Brizo). The authors also add new modules with the OpenStack Block Storage (Cinder) and Nova Compute to communicate with the new components. OpenStack Dashboard (Horizon) and Identity manager (Keystone) are augmented to provide user interface (UI) and authentication to the proposed components. Finally, they designed a forensics-enabled image for VMs to support the forensics related features. By incorporating the proposed architecture, cloud providers may attract more customers with the assurance of reliable forensics support. Customers also do not need to establish privately owned, costly forensics-enabled computing/storage infrastructures for critical business applications. Therefore, Zawoad *et al.* designed FECloud, in order to preserve the trustworthiness of evidence. On the other hand, the proposed solution by Zawoad *et al.* does not include adequate security measures for data collection and data exporting out of the cloud. Thus, we assume that the collection of evidence could be targeted by different attacks by any intruder in the cloud. Another issue with the authors design where they collect their digital forensics data in one repository named Clio. This design suffers from single point of failure in case the data in Clio is destroyed or no communication with Clio is available.

In a more recent work, Hou *et al.* [11] presented a privacy-preserving approach for collecting evidence in forensic investigation. As the servers under investigation may store data from thousands or even hundreds of thousands of innocent. The investigator may have no right to access the irrelevant data. In particular, some of them may involve confidential information. A simple solution would be to ask the server administrator to retrieve only the data relevant to the crime (or the suspect) and hand this to the investigator. This simple solution also may not work since the investigator may not want the administrator to know what he is looking for due to the specific nature of the crimes. The authors assume that the server administrator is willing to cooperate, search the relevant data, and return all located data to the investigator whenever a warrant is provided. They assume that their solutions can satisfy the following security goals: Privacy, Confidentiality, Integrity/Authenticity, Efficiency, and Multiple investigators. In [11], the investigator specifies single or multiple keyword(s) based on investigation subject, encrypts and sends it (or them) to the server administrator. Then, the administrator encrypts all the data files stored on the server (where each data file is represented as a set of words), searches for encrypted keyword(s) in the encrypted data files and returns the relevant data (i.e., the data files containing the keyword(s)) to the investigator. By searching the encrypted data files with encrypted keyword(s), the administrator has no idea of what keyword(s) the investigator is looking for; by performing investigation only on the relevant data files returned by the administrator, the investigator has information about other irrelevant data files (i.e., the data files that do not contain the keyword(s)). It should be pointed out that while not perfect; keyword searching is currently the most widely recognized culling method in the area of digital forensics and e-discovery. The schemes for single keyword search on encrypted data are based on homomorphic encryption and commutative encryption, and the schemes for multiple keyword search are based on the protocol for privacy preserving set intersection. These schemes can satisfy privacy and confidentiality requirements. However, these schemes utilize encryption technology directly or indirectly, so the efficiency may be a concern due to the time consuming encryption and decryption procedures on large amounts of data. Moreover, these schemes do not satisfy the Integrity/Authenticity. Questionable Integrity/Authenticity of data is not admissible in courts. These existing schemes can only handle multiple investigators by encrypting the data for each investigator. To tackle this problem, the authors propose the use of (t,n) threshold secret sharing schemes and their homomorphism properties to improve the investigation efficiency and verify the data integrity and authenticity. The proposed solution in Hou *et al.*' more recent work [11] is based on searching predefined word list for a certain files in the evidence, which is not practically the target of digital forensics. In digital forensics, we might search the evidence several times; modify the word list depending on the requirements of the case and the findings that may change a lot. The claimed integrity and authenticity of the evidence here is doubtful. The administrator only searches for evidence in a defined collected file by himself. The mechanism for how to transfer the collected evidence to the investigator is not presented in a secure way. Another important thing is the added layer of a third party for comparing the word list with the evidence files, should more layer of trust and authenticity be required, which is not clarified from the authors.

Authentication protocols are the basis of security in many distributed systems, and it is therefore essential to ensure that these protocols function correctly. Unfortunately, their design has been extremely error prone. Most of the protocols found in the literature contain redundancies or security flaws. In [4], Nasreldin *et al.*, show that Hou *et al.*' scheme [3] does not preserve its claimed integrity and authenticity. In [3], Hou *et al.* proposed an "encryption-then-blind signature with designated verifier" scheme to prove the authenticity and integrity of the evidence in cloud environment. In [4], Nasreldin *et al.* show that the encrypt-then-sign scenario has a potential pitfall. Assume that Alice has discovered a breakthrough business idea and wants to inform her boss, Victor, about her discovery. Then, Alice will encrypt the message M using Victor's public key and then sign the result using her secret key. Next, Alice sends $[\{M\}_{Victor}]_{Alice}$ to Victor. However, Bob can set himself as a man-in-the middle and intercept messages from Alice to Victor. Bob can then use Alice's public key to compute $\{M\}_{Victor}$. Then, Bob signs it and sends $[\{M\}_{Victor}]_{Bob}$ to Victor. When Victor receives $[\{M\}_{Victor}]_{Bob}$ and verifies Bob's signature on it, Victor will assume that Bob has made this astonishing discovery and Alice cannot disprove Bob's claim, as shown in Figure 1 and Figure 2.

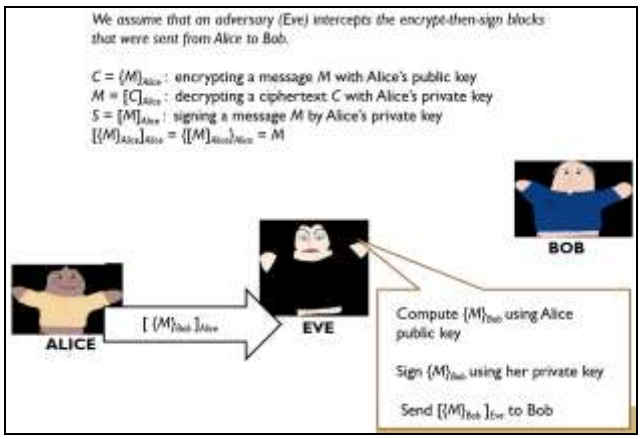

Figure 1: Cryptanalysis - Step 1

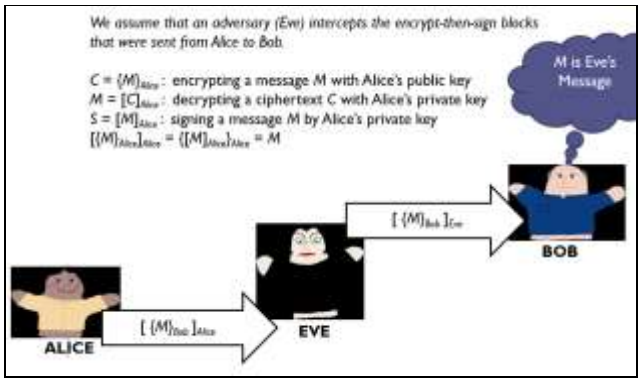

Figure 2: Cryptanalysis - Step 2

Nasreldin et al.[4] presented a modification to Hou et al.' scheme [3] in order to overcome the discovered security pitfalls by using three-block approach (i.e. Sign-Encrypt-Sign and Encrypt-Sign-Encrypt.). Confidentiality, authenticity, and integrity goals can be achieved through cryptographic algorithms. Information security aims to protect the availability, privacy, and integrity of data through the use of digital signature and encryption algorithms. Data confidentiality and data integrity are two of the most important functions of modern cryptography. There are two possible scenarios: Sign-Encrypt-Sign and Encrypt-Sign-Encrypt. Both scenarios can resist the cipher-text forwarding attack and its consequences. To ensure confidentiality and chain of custody for the digital forensics process in the cloud, Hou et al.' scheme (section 3.2 in [3]) needs another encryption step, at the sender side, after the "Blind signature" step (step number 2 in section 3.2 in [3]). At the recipient side, Hou et al.' scheme needs another decryption step after the "Signature verification" step (step number 4 in section 3.2 in [3]). These are two extra steps (one block for encryption at the sender side and one block for decryption at the recipient side) are the mitigations for Hou et al.' scheme against the plaintext-subsection and cipher text stealing attacks.

Nasreldin *et al.* [12] propose an identity-based signcryption protocol to reduce the computation, communication, and implementation overheads in evidence collecting in cloud forensics. Their proposed protocol is more efficient than all the previously presented protocols. It allows the recipient (verifier) to restore the message blocks upon receiving their corresponding signature blocks. The proposed protocol is perfect for some application requirements and it fits packet switched networks. The proposed protocol has two stages of verification to ensure that the message has been recovered efficiently and correctly. The first verification step is to ensure the integrity and authenticity of the message (e.g., no modification or substitution in the ciphertext r_i). The second verification step is to ensure that the message M_i is reconstructed successfully. This stage is useful for public verification in the case of a dispute takes place. It guarantees that the proposed protocol satisfies the non-repudiation property. Nasreldin *et al.*'s details [12] are in Figure 3, Figure 4, Figure 5, and Figure 6.

Nasreldin *et al.* [12] show that the security of the proposed protocol is based on the intractability of reversing the secure cryptographic hash function and the Elliptic Curve Discrete Logarithm (ECDL) problem. Moreover, they analyzed the security of the proposed protocol in terms of authenticity, unforgeability, confidentiality, non-repudiation, and forward secrecy.

Setup:

The Private Key Generation center (PKG) chooses:

- *Gap Diffie-Hellman* group G_1 of prime order q
- Multiplicative group G_2 of the same order
- Bilinear map $e : G_1 \times G_1 \rightarrow G_2$
- Arbitrary generator $P \in G_1$
- Random value $s \in Z_q^*$ as the master secret key
- Secure cryptographic hash function $H_1 : 0, 1^* \rightarrow G_1$
- Secure cryptographic hash function $H_2 : 0, 1^* \rightarrow Z_q^*$

The Private Key Generation center (PKG) computes:

- The corresponding public key $P_{pub} = sP$

The system parameters $(G_1, G_2, P, P_{pub}, H_1, H_2, e, q)$ and the master secret key is s

Figure 3: Setup - New Secure Communication Design for Digital Forensics [12]

KeyExtract:

Given identity ID, PKG:

- Compute $S_{ID} = sH_1(ID)$
- Send S_{ID} to the user with identity ID
- Define Q_{ID} as the public key of the user with identity ID
- Choose arbitrary generator $P \in G_1$
- Choose random value $s \in Z_q^*$ as the master secret key
- Choose secure cryptographic hash function $H_1: 0, 1^* \rightarrow G_1$
- Choose secure cryptographic hash function $H_2: 0, 1^* \rightarrow Z_q^*$

The sender A has public key $Q_A = H_1(ID_A)$ and secret key $S_A = sQ_A$

The recipient B has public key $Q_B = H_1(ID_B)$ and secret key $S_B = sQ_B$.

The sender A divides the stream message M into blocks, M_i, where $M_i \in Z_q^*$ in order to send it to the recipient B.

Figure 4: KeyExtract - New Secure Communication Design for Digital Forensics [12]

Signcrypt:

The sender A, with secret key S_A and public key Q_A, uses the following steps before sending the signcrypted message. The sender A chooses:

- Random number $k \in Z_q^*$
- Lets $r_0 = 0$.

Then, A calculates:

- $r_i = M_i \cdot H_2(r_{i-1} \oplus e(P, Q_B)^k)$, for $i = 1, 2, 3, \ldots, n$
- $\alpha = H_2(r_1, \ldots, r_n, e(P, Q_B)^k)$
- $\beta = H_2(M_1, \ldots, M_n, \alpha, e(P, P)^k)$
- $\gamma = \beta \cdot P$
- $\theta = \beta \cdot Q_B$
- $S = \beta^{-1} \cdot k \cdot P - \beta^{-1} \cdot S_A$

The sender, A, $(S, \alpha, \gamma, \theta, r_1, \ldots, r_n)$ The recipient, B

non-secure channel

Figure 5: SignCrypt - New Secure Communication Design for Digital Forensics [12]

II. RSA vs. ECC Asymetric Cryptosystems

In this section, we show many comparisons on different platforms demonstrating that the proposed modification to Hou *et al.* (Encrypt-Sign-Encrypt) has lower execution times over ECC than RSA, TABLE I.

Cryptography is used to transmit the data securely in open networks. The Elliptic Curve Cryptosystem (ECC) is used in many applications. ECC cryptosystem is better future option than RSA and discrete logarithm systems. For this reason ECC is an excellent choice for doing asymmetric cryptography in portable devices right now. The smaller ECC keys in turn makes the cryptographic operations that must be performed by the communicating devices that are to be embedded into considerably smaller hardware. Therefore, software applications may complete cryptographic operations with fewer processor cycles, and operations can be performed much faster, while still retaining equivalent security. This means, in turn, reduced power consumption, less space consumed on the printed circuit board, and software applications that run more rapidly creating lower memory demands. ECC has faster computations, reduced power consumption, as well as savings in memory space and bandwidth. In brief, for communication using smaller devices and asymmetric cryptosystem, we need ECC. The contrast in key lengths of RSA, Symmetric cryptosystems, and ECC are shown in TABLE II and Figure 7. Private keys are 12-times larger for RSA compared to ECC at the 128-bit security level, TABLE III. TABLE IV shows us the energy cost of RSA and ECC.

ECC holds out the promise of a more cost efficient method to perform encryption and to secure transmission over internet. Elliptic curves are believed to provide good security with smaller key sizes, something that is very useful in many applications. Smaller key sizes may result in faster execution timings for the schemes, which is beneficial to systems where real time performance is a critical factor.

TABLE I. ECC AND RSA (ADVANTAGES/DISADVANTAGES)

	ECC	**RSA**
Advantages	• Smaller keys, ciphertexts and signatures. • Very fast key generation. • Fast signatures, moderately fast encryption/ decryption. • Signatures can be computed in two stages. • Good protocols for authenticated key exchange. • Better US government support. • Special curves with bilinear pairings allow new-fangled crypto.	• Fast, very simple encryption and verification. • Easier to implement than ECC. • Easier to understand. • Signing and decryption are similar; encryption and verification are similar. •Widely deployed, better industry support.
Disadvantages	• Complicated and tricky to implement securely. • Standards aren't state-of-the-art. • Signing with a broken random number generator compromises the key. • Still has some patent problems. • Newer algorithms could theoretically have unknown weaknesses.	• Very slow key generation. • Slow signing and decryption, which are slightly tricky to implement securely. • Two-part key is vulnerable to Grater common Divisor (GCD) attack if poorly implemented.

Due to the security issues, most new cryptographic protocols are moving away from RSA to elliptic curves. That transition is happening even faster in the embedded space where the ECC cost/performance benefits quickly become significant.

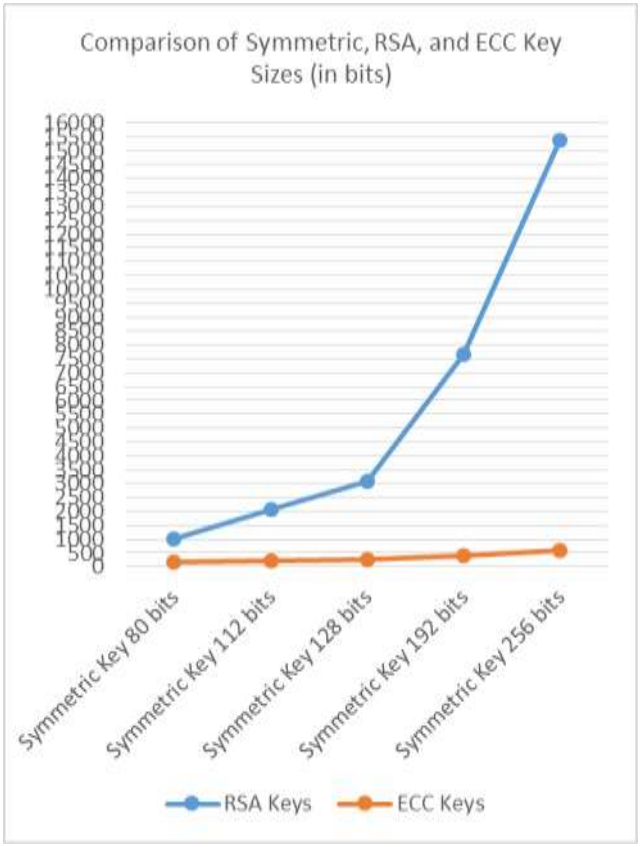

Figure 7: Comparison of Symmetric, RSA, and ECC Key Sizes

In TABLE V, TABLE VI, and Figure 8, we give some sample timings for RSA and ECC on different platforms. In TABLE VI, RSAd is the private key operation, whereas RSAe is the public key operation. Figure 8 shows that the 163-bit ECC/1024-bit RSA security level, an elliptic curve exponentiation for general curves over arbitrary prime fields is roughly 5 to 15 times as fast as an RSA private key operation, depending on the platform and optimizations. At the 256-bit ECC/3072-bit RSA security level the ratio has already increased to between 20 and 60, depending on optimizations. To secure a 256-bit AES key, ECC-521 can be expected to be on average 400 times faster than 15,360-bit RSA. As seen in TABLE II, ECC (at the current minimum 1024-bit security level) has the computation time for the signer per message as 22.9 ms instead of 188.7 ms in RSA case, TABLE VI (on StrongARM). Strong public-key cryptography is often considered to be too computationally expensive if not accelerated by cryptographic hardware. The proposed modification by Nasreldin *et al.* [4] contemplates an Encrypt-Sign-Encrypt structure that could be implemented though RSA or ECC cryptosystems. TABLE I illustrates that ECC has more advantages over RSA. Moreover, TABLE VII elaborates on the execution times of Encrypt-Then-Sign and Encrypt-Sign-Encrypt on the Ultra SPARC II platform. Figure 9 shows the comparison between execution times of the non-secure Hou *et al.* scheme, the proposed modification based on RSA-Encrypt-Then-Sign, and the proposed modification based on the ECC-Encrypt-Sign-Encrypt on the Ultra SPARC II platform. Furthermore, TABLE VIII shows the execution times of Encrypt-Then-Sign and Encrypt-Sign-Encrypt on StrongARM platform. Figure 10 shows the comparison between execution times of non-secure Hou *et al.* scheme, proposed modification based on RSA-Encrypt-Then-Sign, and proposed modification based on ECC-Encrypt-Sign-Encrypt on StrongARM platform. All comparisons between the two proposed modifications (using Encrypt-Sign-Encrypt instead of Encrypt-Sign as Hou *et al.* suggested) illustrate that ECC has less key size and less computation time than RSA on all the selected platforms.

TABLE IX shows that the new secure communication design for digital forensics [13] outperforms the proposed Encrypt-Sign-Encrypt (ECC) [4] in terms of low computation cost. Also, TABLE X shows a comparison between evidence acquisition protocols.

TABLE II. KEY SIZES FOR EQUIVALENT SECURITY LEVELS (IN BITS), [13-19]

Security bits (Key Size)	**Symmetric Encryption Algorithms**	**Minimum Size (bits) of Public Keys**	
		RSA Key size (bits)	Elliptic Curve Key Size (bits)
80	Skipjack	1024	163
112	3DES	2048	224
128	AES-128	3072	283
192	AES-192	7680	384
256	AES-256	15360	571

TABLE III. ECC AND RSA KEY SIZE RATIO, [13-19]

ECC key size	163	283	384	571
RSA key size	1024	3072	7680	15360
Key size ratio	1 : 6	1 : 11	1 : 20	1 : 30

TABLE IV. ENERGY COST OF DIGITAL SIGNATURE AND KEY EXCHANGE COMPUTATIONS [19]

Algorithm	**Signature**		**Key Exchange**	
	Sign	Verify	Client	Server
RSA 1024	304	11.9	15.4	304
ECC 163	22.82	45.09	22.3	22.3
RSA 2048	2302.7	53.7	57.2	2302.7
ECC 224	61.54	121.98	60.4	60.4

TABLE V. ELLIPTIC CURVE EXPONENTIATION TIMING (MS), [15]

Processor	**MHz**	**163-bit**	**224-bit**
Ultra SPARC II	450	6.1	8.7
StrongARM	200	22.9	37.7

TABLE VI. RSA ENCRYPT/DECRYPT TIMINGS (MS), [15]

Processor	MHz	1024-RSA_d	1024-RSA_e	2048-RSA_d	2048-RSA_e
Ultra SPARC II	450	32.1	1.7	205.5	6.1
StrongARM	200	188.7	10.8	1273.8	39.1

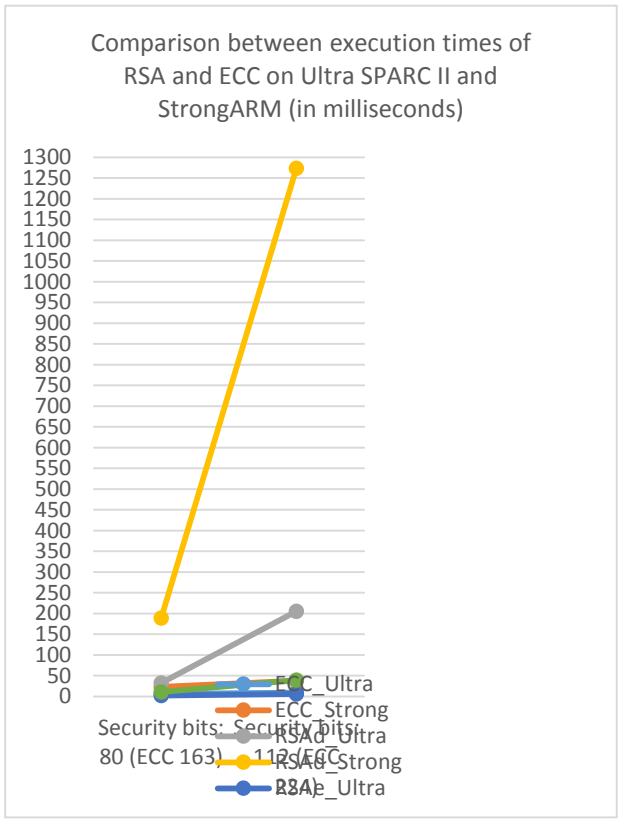

Figure 8: Comparison between execution times of RSA and ECC on Ultra SPARC II and StrongARM (in milliseconds)

TABLE VII. EXECUTION TIMES OF ENCRYPT-THEN-SIGN AND ENCRYPT-SIGN-ENCRYPT ON THE ULTRA SPARC II (IN MILLISECOND)

Key Size in bits		Non-secure Hou *et al.* scheme Encrypt-Then-Sign	Proposed modification to Hou *et al.* scheme based on RSA Encrypt-Sign-Encrypt	Proposed modification to Hou *et al.* scheme based on ECC Encrypt-Sign-Encrypt
RSA	**ECC**			
1024	**163**	2(1.7+32.1) = 67.6	3(1.7+32.1) = 101.4	6*6.1 = 36.6
2048	**224**	2(6.1+205.5) = 423.2	3(6.1+205.5) = 634.8	6*8.7 = 52.2

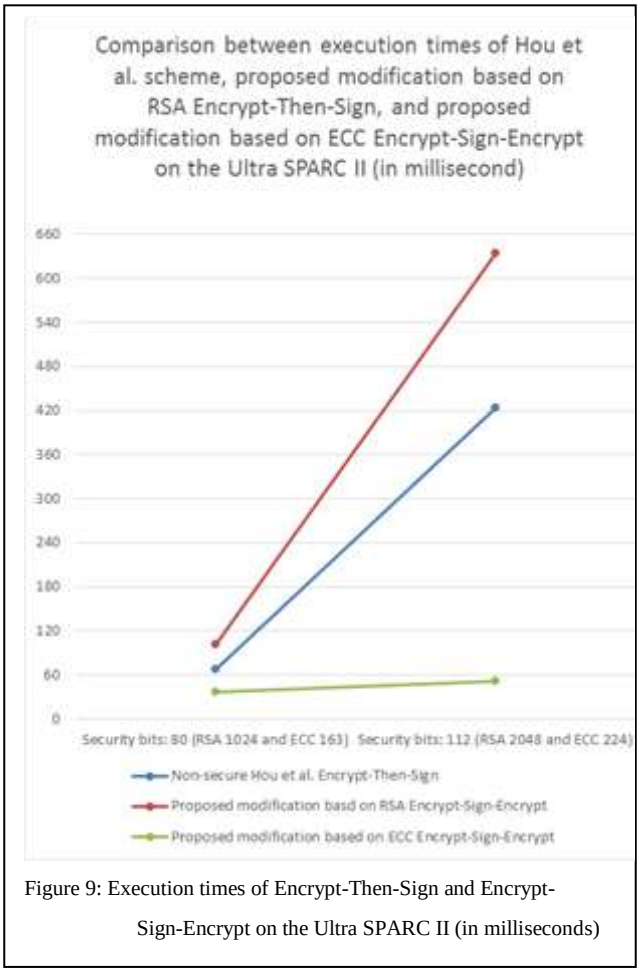

Figure 9: Execution times of Encrypt-Then-Sign and Encrypt-Sign-Encrypt on the Ultra SPARC II (in milliseconds)

TABLE VIII. EXECUTION TIMES OF ENCRYPT-THEN-SIGN AND ENCRYPT-SIGN-ENCRYPT ON THE STRONGARM (IN MILLISECOND)

Key Size in bits		Non-secure Hou *et al.* scheme Encrypt-Then-Sign	Proposed modification to Hou *et al.* scheme based on RSA Encrypt-Sign-Encrypt	Proposed modification to Hou *et al.* scheme based on ECC Encrypt-Sign-Encrypt
RSA	**ECC**			
1024	**163**	2(10.8+188.7) = 399	3(10.8+188.7) = 598.5	6*22.9 = 137.4
2048	**224**	2(39.1+1273.8) = 2625.8	3(39.1+1273.8) = 3938.7	6*37.7 = 226.2

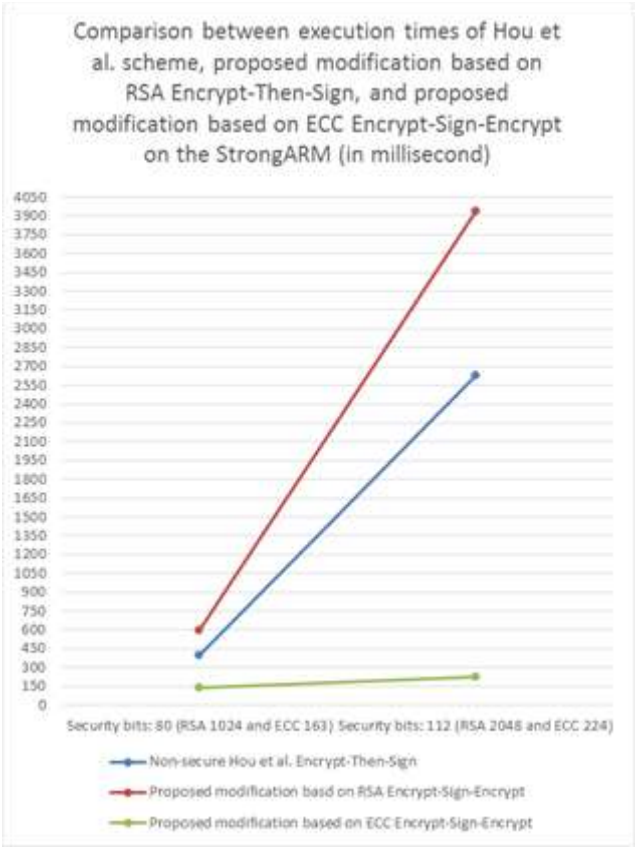

Figure 10: Execution times of Encrypt-Then-Sign and Encrypt-Sign-Encrypt on the StrongARM (in milliseconds)

TABLE IX. ARITHMETIC OPERATIONS IN NEW SECURE COMMUNICATION DESIGN FOR DIGITAL FORENSICS VS. ENCRYPT-SIGN-ENCRYPT

	New Secure Communication Design for Digital Forensics	**Encrypt-Sign-Encrypt**
Sender	SignCrypt: 4 point_mul + 1 point_add + 1 mod_mul + 3 hash + 1 xor ≈ 4 point_mul + 1 point_add + 1 mod_mul	Encrypt-Sign-Encrypt: 2 (2 point_mul + 1 point_add) + 1 point_mul + 1 point_add ≈ 5 point_mul + 3 point_add
Recipient	UnSignCrypt 3 hash + 2 bilinear_map ≈ 2 bilinear_map	Decrypt-Verify-Decrypt: 2 (1 point_mul + 1 point_add) + 2 point_mul + 1 point_add
Total	4 point_mul + 1 point_add + 1 mod_mul + 2 bilinear_map	9 point_mul + 6 point_add

point_mul: point multiplication over elliptic curve

point_add: point addition over elliptic curve

mod_mul: modular multiplication

bilinear_map: bilinear pairing maps

hash: cryptographic hash function operation

XOR: BITWISE XOR LOGIC OPERATION

TABLE X. COMPARISON BETWEEN EVIDENCE ACQUISITION PROTOCOLS

Protocol	Digital Forensics in the Cloud [112-116]	A Privacy-Preserving Approach for Collecting Evidence in Forensic Investigation [117]	Verifying Data Authenticity and Integrity in Server-Aided Confidential Forensic Investigation [6]	Proposed Encrypt-Sign-Encrypt [102]	Proposed new Secure Communication Design for Digital Forensics [125]
Support Data Blocks	No	No	No	No	Yes
Receiving data out of order	No	No	No	No	Yes
Latency	Yes	Yes	Yes	Yes	No
Authenticity	Yes	Yes	No (claimed Yes)	Yes	Yes
Confidentiality	No	No (keyword only)	Yes	Yes	Yes
Integrity	Yes	Yes	No (claimed Yes)	Yes	Yes
Non-Repudiation	No	No	No	Yes	Yes
Cloud compatibility	Yes	Yes	Yes	Yes	Yes
Secure connection for data collection and transfer	No	No	Yes	Yes	Yes
Developing New Cryptographic Scheme	No	No	No	No	Yes

III. CONCLUSION

In this paper, we have presented a performance comparison between different digital evidence acquisition protocols in the cloud-computing environment. We have also illustrated that some of these protocols do not achieve the claimed security goals. The performance analysis shows that ECC based protocols have less execution time than RSA based protocols.

References

[1] S. Hou, T. Uehara, S.M. Yiu, L.C.K. Hui, and K.P. Chow, "Privacy Preserving Confidential Forensic Investigation for Shared or Remote Servers," Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 378-383, 2011.

[2] S. Hou, T. Uehara, S.M. Yiu, L.C.K. Hui, and K.P. Chow, "Privacy Preserving Multiple Keyword Search for Confidential Investigation of Remote Forensics," Third International Conference on Multimedia Information Networking and Security, pp. 595-599, 2011.

[3] S. Hou1, R. Sasaki, T. Uehara, and S. Yiu, "Verifying Data Authenticity and Integrity in Server-Aided Confidential Forensic Investigation," Lecture Notes in Computer Science 7804, Springer, pp. 312-317, 2011.

[4] M. Nasreldin, M. El-Hennawy, H. Aslan, and A. El-Hennawy, "Digital Forensics Evidence Acquisition and Chain of Custody in Cloud Computing," International Journal of Computer Science Issues, vol. 12, issue 1, no. 1, 2015, pp. 153-160.

[5] Y. Zheng, "Digital signcryption or how to achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption)," Proc. of CRYPTO'97, LNCS 1294, pp. 165179, Springer-Verlag, 1997.

[6] S. Zawoad and R. Hasan, "Digital Forensics in the Cloud", Journal of Defense Software Engineering, vol. 26, no. 5, pp. 17-20, 2013

[7] S. Zawoad and R. Hasan, "FECloud: A Trustworthy Forensics-Enabled Cloud Architecture", 11th Annual IFIP WG 11.9 International Conference on Digital Forensics, Orlando, Florida, January 2015.

[8] S. Zawoad, A. Dutta, and R. Hasan, "Towards Building Forensics Enabled Cloud Through Secure Logging-as-a-Service", IEEE Transactions on Dependable and Secure Computing (TDSC), SI-Cyber Crime, vol. -, no. -, pp. -, 2015.

[9] S. Zawoad and R. Hasan, "Towards a Systematic Analysis of Challenges and Issues in Secure Mobile Cloud Forensics", The 3rd International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco (IEEE Mobile Cloud), March 2015.

[10] S. Zawoad, R. Hasan, and J. Grimes, "LINCS: Towards building a trustworthy litigation hold enabled cloud storage system," Digital Investigation, Elsevier, vol. 14, S55-S67, 2015.

[11] S. Hou, Y. Siu-Ming, U. Tetsutaro, and S. Ryoichi, "A Privacy-Preserving Approach for Collecting Evidence in Forensic Investigation," International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 2, no. 1, pp. 70-78, 2013.

[12] M. Nasreldin, M. El-Hennawy, H. Aslan, and A. El-Hennawy, "New Secure Communication Design for Digital Forensics in Cloud Computing," International Journal of Computer Science and Information Security (IJCSIS), vol. 13, no. 2, pp. 8-17, 2015.

[13] https://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[14] https://www.nsa.gov/ia/programs/suiteb_cryptography/

[15] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," IEEE Wireless communications, vol. 11, no. 1, pp. 62-67, 2004.

[16] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," In Cryptographic hardware and embedded systems-CHES, pp. 119-132. Springer Berlin Heidelberg, 2004.

[17] R. Battistoni, R Di Pietro, F Lombardi, "CloRoFor: Cloud Robust Forensics," arXiv preprint arXiv:1506.01739, (2015).

[18] RSA vs ECC Comparison for Embedded Systems https://www.atmel.com/.../Atmel-8951A-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper_072015

[19] N. Thiranant, Y.-S. Lee, and H.-J. Lee, "Performance Comparison between RSA and Elliptic Curve Cryptography-Based QR Code Authentication," IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp.278-282, March 2015.