FALSE BOTTOM ENCRYPTION: DENIABLE ENCRYPTION FROM SECRET SHARING

Humera Masood[1], Hafsa Khatoon, Dr. K. Palani
[1]PG Scholar, Department of Information Technology, Shadan Women's College of Engineering and Technology, Hyderabad, Humeramaaood05@gmail.com
[2]Assistant Professor, Department of Computer Science and Engineering, Shadan Women's College of Engineering and Technology, khafsakhan12@gmail.com
3Professor, Department of Computer Science and Engineering, Shadan Women's College of Engineering and Technology, Hyderabad, India, principalswcet2020@gmail.com

ABSTRACT

We demonstrate how to use secret sharing to construct a deniable encryption technique. We can avoid both computational intractability assumptions and data preprocessing, in contrast to the related concept of honey encryption, which uses a preprocessing step in symmetric encryption to reshape the distribution of a plaintext towards making the real plaintext indistinguishable from a cipher text for a fake message. This achieves deniability against an attacker with sufficient computing power to compel decryptions and brute-force break a cipher text. In accordance with the idea of plausible deniability, we have many decryption keys to reveal separate plaintexts contained within the same encrypted text. For example, an attacker will be persuaded that a plaintext extracted from a cipher text using a key that a victim divulged under duress is authentic, even while the true secret is still hidden. Inauthentic Bottom Using the same key for both encryption and decryption, encryption creates a symmetric technique that combines the features of deniable encryption and honeypot encryption. As a security feature, we explicitly define and distinguish "deniable" from "plausibly deniable," demonstrating how, depending on the plaintext distribution, plausible deniability reverts to (only) deniability. Our method, which is based on secret sharing, is easy to use, lightweight, and effective in both encryption and decryption. We do not, therefore, depend on computational intractability.

1. INTRODUCTION

Generally speaking, a sender and receiver must first share information to communicate safely. In many cases, encryption and good password protection may be sufficient to safeguard your data. For instance, with AES, the sender and receiver share the same key in a symmetric encryption system. However, using the RSA technique, the sender and receiver of an asymmetric-key encryption technique exchange a public system parameter and the recipient's public key, with the public key delivery occurring through public key infrastructure. These general encryption techniques offer a security guarantee against eavesdropping attempts, but they fall short when faced with threats of coercion. Even if the attacker does not have access to the key, if it intercepts the cipher text, it may be able to force both the sender and the receiver to decrypt the message. Non-committing encryption [1] and deniable encryption [2] have been presented as solutions to this issue. Users can decrypt an existing cipher text associated with a certain counterfeit message using these two different encryption algorithms. The first algorithm is called the sender's encryption algorithm to encrypt a message under a secret key sk. The second algorithm, known as the faking algorithm, is publicly known, and the sender uses this fake algorithm to produce fake messages. Getting the same cipher text from two different algorithms is computationally cumbersome. In our work, we show how the actual message and the fake message cipher texts can both be produced using a single algorithm that also is computationally efficient. There are many circumstances under which plausible deniability may also be necessary. If your opponents cannot obtain your password, strong encryption can keep them out. However, if the threat model incorporates coercion, such as the prospect of a jail term or torture, you might give up and hand over the key to rescue yourself. As a result, the attacker would have access to the data, perhaps putting you at risk for later repercussions. The idea of plausible deniability originates in politics and espionage and refers to one's capacity to downplay one's culpability for, or knowledge of specific facts or events. It may entail carrying out operations in a way that leaves no trace, especially changing systems around particular people, to enable them to honestly deny their knowledge of what took place. Destruction of evidence is another method that can be used to make a given action plausible to deny, but there are also positive use-cases as we will outline next. We question whether it is conceivable to produce cipher text that appears to be for certain claimed receivers but are actually for different receivers. Imagine that Alice wants to secretly send her friend John a message. If she encrypts and sends it to John, she could be asked by her mother who the message was for and command her to decrypt the message. Consequently, John might get a call from Alice's mother, asking him to confirm as well what it

says. To prevent this from happening, Alice can encrypt the message using deniable encryption. Alice will first prepare a pair of texts. One is a trivial message for Bob, whereas the other is a simple covert message for John. John's text is jointly encrypted with Bob's text by Alice using a suitable encryption algorithm. Alice posts the cipher text to a public channel and requests her friends to download the message. The only two people who can successfully decrypt the cipher text are Bob and John, but they produce two different messages: the fake message and the real message. The term ''successful decryption'' refers to the fact that Bob and John are able to decrypt and receive useful messages from the sender. Alice can tell her mother that the cipher text is for Bob and reveal the message that was transmitted to Bob when questioned. Considering that Bob only knows what he has received, he can also be a trustworthy witness. Even if Alice's mother thinks something is concealed in the cipher text, she cannot determine which of Alice's friends the true recipient is. In this case, Alice does not need to help John because her mother would not be able to suspect John, unless she suspects all of Alice's friends.

2. EXISTING SYSTEM OF PROJECT

- However, using the RSA technique, the sender and receiver of an asymmetric-key encryption technique exchange a public system
- In our work, we show how the actual message and the fake message cipher texts can both be produced using a single algorithm that also is computationally efficient.
- The second algorithm, known as the faking algorithm, is publicly known, and the sender uses this fake algorithm to produce fake messages. Getting the same cipher text from two different algorithms is computationally cumbersome.
- The first algorithm is called the sender's encryption algorithm to encrypt a message under a secret key sk.
- Users can decrypt an existing cipher text associated with a certain counterfeit message using these two different encryption algorithms.

3. RELATED WORK

The concept of honey encryption [3] is a generic construction to extend conventional encryption by making decryptions under the wrong key ''appear to be plausible''. This is accomplished by transforming the input plaintext towards obtaining a certain fixed distribution that matches the distribution of the decryption result under a different key. Consequently, if the (same) cipher text is decrypted under the real or the fake key, the resulting plaintexts (one real, the other being fake) will have approximately the same distribution. The security of honey encryption relies on the probability of an attacker judging a plaintext to be legitimate can be calculated by the encrypting party at the time of encryption. The main difference to deniable encryption and to our scheme is that honey encryption does not insert a second plaintext into the cipher text. As in conventional encryption, there still is only one plaintext inside the cipher text, but false decryptions should become less recognizable. This approach aims at retaining security even against keys or plaintexts of low min-entropy, which can be efficient to guess. The main difference to deniable encryption and our scheme is thus in the attacker model: we assume (as does deniable encryption) that the attacker puts force on the plaintext owner to open the cipher text, while honey encryption lets the attacker attempt decryptions under keys of its own (random) choice. The construction of honey encryption makes use of distribution-transforming encoders that aim to shape the distribution of a random plaintext towards a desired and fixed target distribution. Our scheme can use such encoders as well, as a source of plausibly looking plaintexts to act as fakes. We will not make explicit use of such transformations, but mention them as a possible technical implementation of our assumption that fake plaintexts are producible with the same distribution as the real secret plaintexts. Canetti et al. [1] initially developed the concept of deniable encryption. A deniable shared key scheme and a public key scheme are two types of deniable encryption. A straight forward illustration of deniable encryption is the one-time pad: Let m be the original message to be encrypted, and c be the cipher text such that c = m Lk where k represents the shared key. Nobody can refute the encrypt or's assertion that the message is m ′ using the key k ′ = m ′ Lc. In Canetti et al. [2] technique, falsified messages with strong justification were presented using the idea of a translucent set: roughly speaking, this is a set whose membership is not decidable efficiently without trapdoor information. Encryption of a bit b is done by emitting a random string if b = 0 or a string from the translucent set T if b = 1. Under duress, the plaintext bit is deniable, since the claim of having taken a random string or one from T is not efficiently verifiable without the trapdoor information to decide its membership in T . According to Canetti et al., this system is sender-deniable, meaning that the sender can produce proof of falsified messages. Canetti et al. also extended the scheme through an interactive approach, to support receiver-deniability and combined them into a bi-deniable encryption scheme. Numerous researchers have constructed translucent sets using a variety of methods based on this concept. Samplable encryption was implemented by Dürmuth and Freeman [4] to create a translucent set. A bi-translucent set built on a lattice was created by O'Neill et al. [5], in which they emphasize that the schemes are no interactive and

involve no third parties as they build bi-deniable public-key cryptosystems that allow both the sender and the recipient to communicate simultaneously.

4. PROPOSED SYSTEM

- In contrast, the related idea of honey encryption uses a preprocessing phase in symmetric encryption to reshape a plaintext's distribution so that the true plaintext and cipher text data are identical.
- For example, in a symmetric encryption system, the sender and the recipient share the same key when using AES.
- Deniable encryption comes in two flavors: a public key scheme and a shared key scheme. The one-time pad provides a simple example of deniable encryption: Let m represent the original message that has to be encrypted.
- Similar to our aspirations, ambiguous multi-symmetric cryptography has been developed in the field of symmetric encryption. They address a number of attack scenarios, such as those involving selected cipher text, known-plaintext, and others.

5. MODULES

1. User Interface Design

We create the project's windows in this module. All users can securely log in using these windows. Users can only connect to the server by providing their login and password in order to establish a connection. The user can log in straight to the server if they have previously left; otherwise, they must register their information, including their email address, password, and username. In order to maintain the upload and download rates, the server will create an account for each user. The user ID will be set to name. Typically, logging in allows access to a certain page.

2. Authority Server

A server for authenticators is the initial module. The server must first create an account and enter their password. The key request is from the authority server. There is a key generator on the authority server. There is a request on the authority server. It was going to contain user requests from the database as well as owner requests.

3. Owner

the third section. Owners can access their register by logging in with authorization from the authority server. The authority server requires a user ID and password to access. The owner has a data store uploaded. Data that has been uploaded is viewable. Data sharing is approved by the owner.

4. User

A user is the fourth module. The user has a password-protected register with an ID. The authority center grants permissions for the user to log in. After adding a user, the authority server must log in. The user can search through data.

5. CSP(Cloud Service Provider)

Cloud has a module that is five.A user ID and password are required to access CSP.CSP has data that is stored in the database.It is possible for CSP to have both owner and user details. A database's stores

6. TECHNIQUES USED IN PROJECT

6.1. Honey Encryption

A Honey Encryption involves repeated decryption with random keys; this is equivalent to picking random plaintexts from the space of all possible plaintexts with a uniform distribution. This is effective because even though the attacker is equally likely to see any given plaintext, most plaintexts are extremely unlikely to be legitimate i.e. the distribution of legitimate plaintexts is non-uniform. Honey encryption defeats such attacks by first transforming the plaintext into a space such that the distribution of legitimate plaintexts is uniform. Thus an attacker guessing keys will see legitimate-looking plaintexts frequently and random-looking plaintexts infrequently. This makes it difficult to determine when the correct key has been guessed. In effect, honey encryption "[serves] up fake data in response to every incorrect guess of the data or encryption key. The security of honey encryption relies on the fact that the probability of an attacker a plaintext to be legitimate can be calculated (by the encrypting party) at the encryption. This makes honey encryption difficult to apply in certain applications e.g. where the space of plaintexts is very large or the distribution of plaintexts is unknown. It also means that honey encryption can be vulnerable if this probability is miscalculated. For example, it is vulnerable to known-plaintext attacks: if the attacker has a crib that a plaintext must match to be legitimate, they will be able to brute-force even Honey Encrypted data if the encryption did not take the crib into account.

6.2. Hash Algorithm

Hashing is mostly made up of three parts:
1. Key: An index or location for an item's storage in a data structure is determined by the hash function, which accepts any string or integer as input.
2. Hash Function: After receiving the input key, the hash function outputs the element's index in an array known as a hash table. The hash index is the name given to the index.
3. Hash Table: A hash table is a type of data structure that uses a unique function known as a hash function to

key values to values. Data is hashed and stored associatively in an array with distinct indexes for each data value.

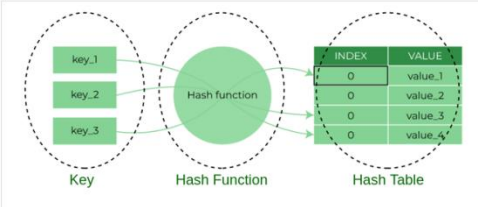

Figure 6.3.1 Hash algorithm Example

Real-Time Applications of Hash Data structure

- Hash is utilized to store data in a cache for quick access. It can also be used to verify passwords.
- Hash functions as a message digest in cryptography.

6.3. SYMMETRIC ALGORITHM USED

Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both the encryption of plaintext and the decryption of cipher text. The keys may be identical, or there may be a simple transformation to go between the two keys.[1] The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link.[2] The requirement that both parties have access to the secret key is one of the main drawbacks of symmetric-key encryption, in comparison to public-key encryption (also known as asymmetric-key encryption). However, symmetric-key encryption algorithms are usually better for bulk encryption. With exception of the one-time pad they have a smaller key size, which means less storage space and faster transmission. Due to this, asymmetric-key encryption is often used to exchange the secret key for symmetric-key encryption.

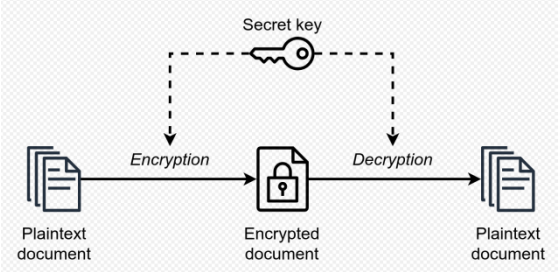

Figure 6.3.2 Symmetric key algorithms

7. DISCUSSION AND CONCLUSION

We have shown a conceptually simple method of concealing information inside an existing sequence of strings, allowing for deceptive decryption in case of forced revelation of decryption keys. At a practical level, matters of storing or remembering the decryption keys have not been discussed, but the use of passwords, for example, is not difficult to imagine here: suppose that whenever we require random values to be chosen, we do so by invoking a pseudorandom number generator (PRNG, e.g., the standardized password-based key derivation function Argon2) that is seeded with a password that the user chooses. In that way, the storage of the value-pair list that constitutes the secret key ski for the message mi , boils down to the choice of a password to open the message mi , from which all random quantities in the process can be recomputed with the password as a seed for a PRNG. The implications to security are, in that sense, to be considered carefully, as the overall entropy about the secret reduces to the min-entropy of the password choice process that determines the hardness of guessing the password, which is the Shannon entropy. Further generalizations may be the inclusion of a third party to establish a four-eyes principle in the opening of a message. That is, for example, one could substitute ρi,1, . . . , ρi,ni by products ρ (a) i,1 • ρ (b) i,1 . . . , ρ (a) i,ni • ρ (b) i,ni , with the individual factors coming from key-bases, respectively root keys, that two persons, Alice and Bob, are given. In that case, an adversary forcing Alice to cooperate would also have to convince Bob to cooperate, in order to discover a meaningful message. As yet another variant, note that the role of the factors from root key r and from c is ‘‘symmetric’’, and hence one could alternate the appending of parts to c with adding parts to the r. Storing c in a remote location and keeping r on one’s own local computer then creates the seeming appeal of putting new information into c without actually letting c visibly grow. This instance of the scheme is, however, not considered as useful here, since it is nothing else than storing an encrypted version of a message locally, and letting the key to this message be stored remotely at a possibly untrusted location Practical room for improvement is in the scheme’s necessity to remember and use all secret keys whenever there is a need to modify messages after they went into the ciphertextc. Abandoning this requirement, the key storage and management requirements fall back to those of a conventional secret key encryption with fixed key sizes. Hence, its ‘‘information theoretic’’ security guarantees are in any case bounded by the size of the secret root key to be guessed, but the brute force complexity is still larger than just guessing the secret key, since the adversary may, except if there is so far only 1 message in c, still be uncertain about which parts of c may have been used to represent the secret message. Therefore, the concept of (plausible) deniability is the added value over brute-force attack resilience. Based on our review of literature on deniable encryption schemes, the user can, in past schemes, come up with only one fake message as a counterpart to defend its secret. In contrast, our scheme allows us to bring more than one fake message to hide a secret. Also, this encryption scheme is editable in the

sense that at any instant in time, we can update our message by changing only one element in the cipher text or by changing the key indices. Furthermore, the scheme gives us the freedom to delete any message encrypted inside the cipher text simply by replacing at least one element from the cipher text with a random number. Deleting a message gives us the flexibility to prevent the user who was earlier accessing that message from doing it again. If the user wants to decrypt the cipher text with the older key, the outcome will undoubtedly dissatisfy him. Consequently, False-Bottom Encryption extends deniable encryption by the functionality of adding, editing and deleting possibly several plaintexts inside the ciphertext. Our security definition does not account for adversaries profiling the access patterns of a user, which calls for additional techniques to either randomize or ‘‘equalize’’ all access sequences. Future work will thus investigate extensions to our scheme by means of private information retrieval or other techniques (see the related work, in particular [22]), to analyze if information-theoretic security remains accomplishable or deteriorates against attackers that profile the (physical) device usage.

8.RESULTS:

fig. 7.1 index page

fig.7.2 owner register and login

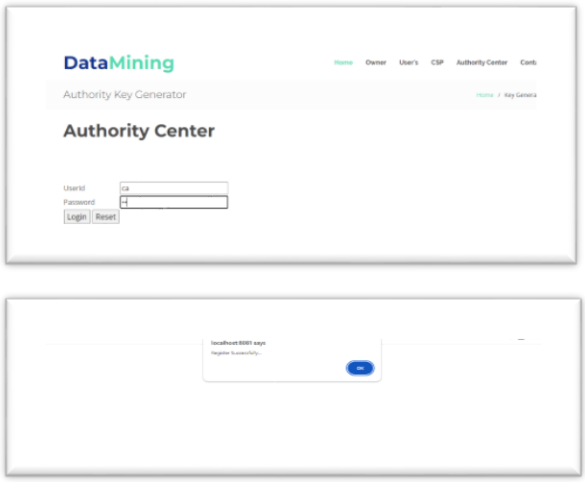

fig. 7.3 the authroity center approves the request for the owner login

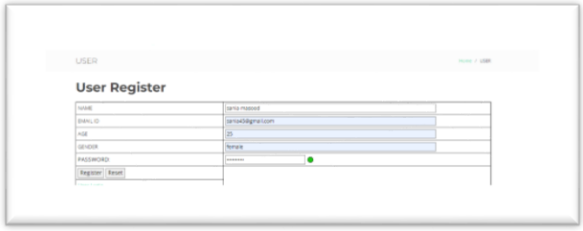

fig.7.4 user registration and login

fig.7.5 user registration and login

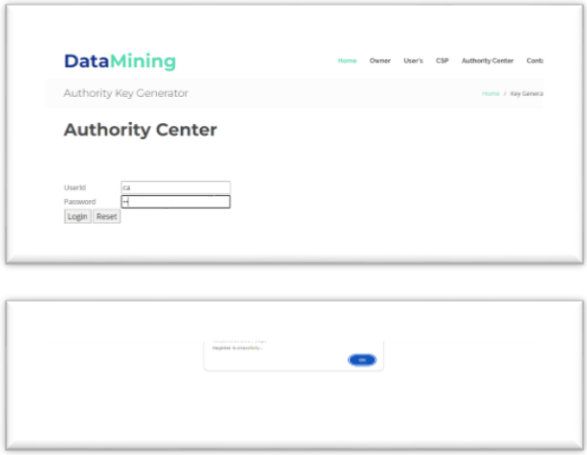

fig.7.6 the authority center approves the request for the user login

fig.7.7 owner/user can upload the file

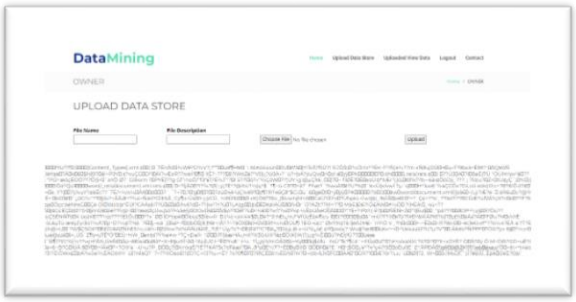

fig.7.8 the uploaded file is encryted in cypher language

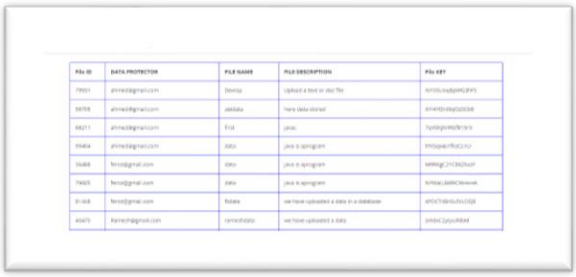

fig. 7.9 the encrpted file generates a hash key

fig. 7.10 the user can access the encrypted data

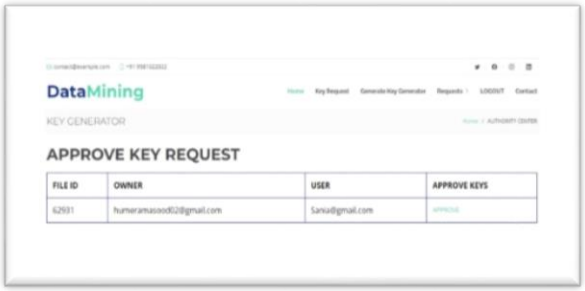

fig .7.11 the user requests for a hash key from the authority center

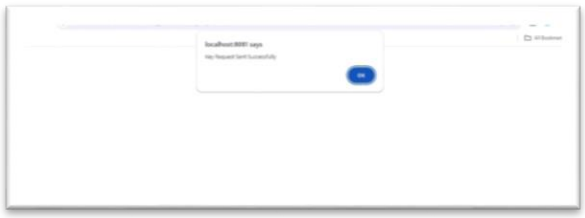

fig. 7.12 the authority center approves the key request

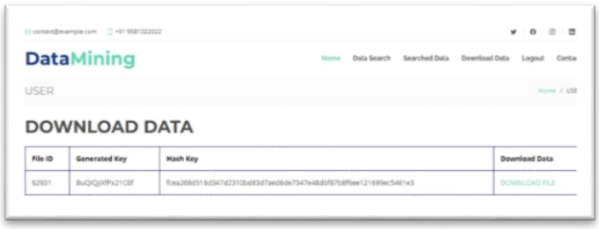

fig. 7.13 the user can download the encrypted file

fig.7.14 the generated hash key is used to decrypt the file

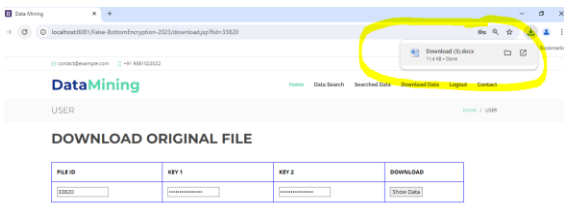

fig. 7.15 the decrypted file can be downloaded and accessed in its original form

FUTURE ENHANCEMENT

Thus, future research will examine ways to expand our scheme using private information retrieval or other methods (refer to the related work, in particular) to examine whether information-theoretic security can still be achieved or if it becomes less effective against attackers who profile the (physical) device usage.

REFERENCES

[1] R. Canetti, U. Feige, O. Goldreich, and M. Naor, ‘‘Adaptively secure multi-party computation,’’ in *Proc. 28th Annu. ACM Symp. Theory Com- put.*, Philadelphia, PA, USA, 1996, pp. 639–648. [Online]. Available: http://portal.acm.org/citation.cfm?doid=237814.238015

[2] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, ‘‘Deniable encryption,’’ in *Proc. 17th Annu. Int. Cryptol. Conf.* (Lecture Notes in Computer Science), vol. 1294. Santa Barbara, CA, USA: Springer, 1997, pp. 90–104.
[3] A. Juels and T. Ristenpart, ‘‘Honey encryption: Security beyond the brute-force bound,’’ in *Advances in Cryptology—EUROCRYPT* (Lec- ture Notes in Computer Science), vol. 8441, D. Hutchison, T. Kanade,
[4] J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, P. Q. Nguyen, and E. Oswald, Eds. Berlin, Germany: Springer, 2014, pp. 293–310, doi: 10.1007/978-3-642-55220-5_17.
[5] M. Durmuth and D. M. Freeman, ‘‘Deniable encryption with negli- gible detection probability: An interactive construction,’’ in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6632, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern,
[6] J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and K. G. Paterson, Eds. Berlin, Germany: Springer, 2011, pp. 610–626, doi: 10.1007/978-3-642-20465-4_33.
[7] A. O’Neill, C. Peikert, and B. Waters, ‘‘Bi-deniable public-key encryption,’’ in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 6841, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg,
[8] F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. P. Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and P. Rogaway, Eds. Berlin, Germany: Springer, 2011, pp. 525–542, doi: 10.1007/978-3-642-22792-9_30.
[9] M. Klonowski, P. Kubiak, and M. Kutylowski, ‘‘Practical deniable encryp- tion,’’ in *SOFSEM 2008: Theory and Practice of Computer Science* (Lecture Notes in Computer Science), V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, Eds. Berlin, Germany: Springer, 2008, pp. 599–609.
[10] P. Gasti, G. Ateniese, and M. Blanton, ‘‘Deniable cloud storage: sharing files via public-key deniability,’’ in *Proc. 9th Annu. ACM Workshop Pri- vacy Electron. Soc.*, Chicago, IL, USA, 2010, p. 31. [Online]. Available:http://portal.acm.org/citation.cfm?doid=1866919. 1866925
[11] M. H. Ibrahim, ‘‘A method for obtaining deniable public-key encryption,’’ *Int. J. Netw. Secur.*, vol. 8, no. 1, pp. 1–9, 2009.
[12] P. Chi and C. Lei, ‘‘Audit-free cloud storage via deniable attribute- based encryption,’’ *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 414–427, Apr. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/7090980/
[13] R. Canetti, S. Park, and O. Poburinnaya, ‘‘Fully deniable interactive encryption,’’ in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 12170, D. Micciancio and T. Ristenpart, Eds. Cham, Switzerland: Springer, 2020, pp. 807–835, doi: 10.1007/978-3-030- 56784-2_27.
[14] C. Dwork, M. Naor, and A. Sahai, ‘‘Concurrent zero-knowledge,’’ *J. ACM*, vol. 51, no. 6, p. 851–898, Nov. 2004, doi: 10.1145/1039488.1039489.
[15] M. Naor, ‘‘Deniable ring authentication,’’ in *Advances in Cryptology— CRYPTO* (Lecture Notes in Computer Science), vol. 2442, G. Goos,
[16] J. Hartmanis, J. van Leeuwen, and M. Yung, Eds. Berlin, Germany: Springer, 2002, pp. 481–498, doi: 10.1007/3-540-45708-9_31.
[17] R. W. Zhu, D. S. Wong, and C. H. Lee, ‘‘Cryptanalysis of a suite of deniable authentication protocols,’’ *IEEE Commun. Lett.*, vol. 10, no. 6, pp. 504–506, Jun. 2006.
[18] A. Fiat and M. Naor, ‘‘Broadcast encryption,’’ in *Proc. Annu. Int. Cryptol. Conf.*, Jan. 1993, pp. 480–491.
[19] J. Li, Y. Wang, Y. Zhang, and J. Han, ‘‘Full verifiability for out- sourced decryption in attribute based encryption,’’ *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 478–487, May 2020. [Online]. Available: https://ieeexplore.ieee.org/document/7936626/
[20] J. Li, Q. Yu, and Y. Zhang, ‘‘Hierarchical attribute based encryption with continuous leakage-resilience,’’ *Inf. Sci.*, vol. 484, pp. 113–134, May 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025519300684
[21] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, ‘‘Flexible and fine-grained attribute-based data storage in cloud computing,’’ *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Sep. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7390098/
[22] S. Reddy, P. S. Reddy, and P. Sravanthi, ‘‘Audit free cloud stor- age via deniable attribute base encryption for protecting user pri- vacy,’’ *Int. J. Sci. Eng. Technol. Res.*, vol. 5, no. 17, pp. 3449–3451, 2016.
[23] R. Bassous, R. Bassous, H. Fu, and Y. Zhu, ‘‘Ambiguous multi-symmetric cryptography,’’ in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 7394–7399.
[24] A. Chakraborti, C. Chen, and R. Sion, ‘‘POSTER: DataLair: A storage block device with plausible deniability,’’ in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1757–1759. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2989061
[25] C. Chen, A. Chakraborti, and R. Sion, ‘‘PD-DM: An efficient locality-preserving block device mapper

with plausible deniability,'' *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 1, pp. 153–171, Jan. 2019. [Online]. Available: https://petsymposium.org/popets/2019/ popets-2019-0009.php

[26] C. Chen, X. Liang, B. Carbunar, and R. Sion, ''SoK: Plausibly deni- able storage,'' *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 2, pp. 132–151, Apr. 2022. [Online]. Available: https://petsymposium. org/popets/2022/popets-2022-0039.php

[27] E.-O. Blass, T. Mayberry, G. Noubir, and K. Onarlioglu, ''Toward robust hidden volumes using write-only oblivious RAM,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 203–214, doi: 10.1145/2660267.2660313.

[28] C. Hargreaves and H. Chivers, ''Detecting hidden encrypted vol- umes,'' in *Communications and Multimedia Security*, B. De Decker and I. Schaumuller-Bichl, Eds. Berlin, Germany: Springer, 2010, pp. 233–244.