such that it uses the CPU's hardware random generator as its randomness source through the `rdrand` instruction. For the public-key signatures, we used the RSA implementation of OpenSSL 1.0.2g with 3072-bit RSA moduli. We embedded this program in an SGX enclave using the Graphene-SGX framework [17].

Performance. We measure the performance of all three protocols on Intel SGX, i.e., of Share, Renew and Reconstruct as described in §3.2. In our experiments, we vary the number of shares n and the reconstructing threshold t and report performance results for documents containing between 100 bytes and 30MBytes of random data. We ran our tests on an Intel i7-7700 CPU clocked at 3.60GHz, with 8GB of RAM and Ubuntu 16.04.5 OS.

Figure 2 shows the time required to perform Share, Renew, and Reconstruct using different values for our parameters, namely $\{n = 3, t = 2\}$, $\{n = 5, t = 3\}$, $\{n = 7, t = 4\}$, $\{n = 9, t = 5\}$, and $\{n = 11, t = 6\}$. In our experiment, the time required to perform the secret-sharing computations scales linearly in the whole document size range. The time required to sign a document scales linearly as well, but there is also a constant-time component of approximately 2ms. Similarly, verifying a signature has a linear component and a constant-time component of approximately 60µs. Creating and renewing shares (Figure 2a, Figure 2b) requires multiple signatures, so the time required does not increase significantly until 3KB, while it increases linearly after 10KB. For a 30MB document, creating the shares takes between 21s and 90s depending on the parameters n and t; renewing the shares takes between 27s and 109s. Reconstructing the shares (Figure 2c) only requires signatures verification, so it increases linearly in the whole range. Reconstructing the shares of a 30MB document takes between 23s and 128s depending on the parameters. To scale with larger document sizes, the document can be segmented into smaller sized document blocks to reduce the total share reconstruction latency. In doing so, the user requesting the total document can receive the reconstructed blocks in a streamlined manner, rather than wait for the complete document.

Our prototype implementation stores the whole document and the shares in enclave memory, which results in a bound on the size of the document in our experiments. Modifying the implementation to stream the shares to the enclave eliminates this bound entirely. We can directly observe that the computation time scales linearly, since the generation and verification of signatures contribute with only a linear overhead to the overall runtime of the protocols.

5 CONCLUSION AND FUTURE WORK

In this paper, we introduced SAFE, a TEE-based long-term secure storage system that provides a significantly more efficient means to protect the confidentiality and integrity of outsourced data. It leverages a TEE for the generation and periodic update of shares of the data and securely provisions the shares to the storage servers. SAFE requires significantly fewer information-theoretically secure point-to-point channels and expensive commitment schemes are no longer required because integrity is provided by signatures. We instantiated the TEE for SAFE using Intel SGX and evaluated the computation runtimes of the protocols. SAFE is TEE-agnostic; any other TEE can be deployed and migration to another TEE is also supported for stronger security and robustness guarantees. Furthermore, SAFE, contrary to previous approaches, does not need to assume a synchronous network.

For future work, we plan to address scenarios where the data has to also be securely processed in a privacy-preserving manner (e.g., studies on genomic data) using secure multi-party computation.

ACKNOWLEDGMENT

This project was co-funded by the Deutsche Forschungsgemeinschaft (DFG) — SFB 1119 CROSSING/236615297, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 850990 PSOTI), the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS), the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

REFERENCES

[1] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. 2009. Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information System Security* 13, 1 (2009), 4:1–4:40.
[2] Charles H. Bennett and Gilles Brassard. 2014. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* 560 (2014), 7–11.
[3] Ferdinand Brasser, Srdjan Capkun, Alexandra Dmitrienko, Tommaso Frassetto, Kari Kostiainen, Urs Müller, and Ahmad-Reza Sadeghi. 2017. DR.SGX: Hardening SGX enclaves against cache attacks with data location randomization. *CoRR* abs/1709.09917 (2017). arXiv:1709.09917 http://arxiv.org/abs/1709.09917
[4] Johannes Buchmann, Ghada Dessouky, Tommaso Frassetto, Ágnes Kiss, Ahmad-Reza Sadeghi, Thomas Schneider, Giulia Traverso, and Shaza Zeitouni. 2020. SAFE: A Secure and Efficient Long-Term Distributed Storage System (Full Version). Cryptology ePrint Archive, Report 2020/690. https://ia.cr/2020/690
[5] Johannes Buchmann, Alexander May, and Ulrich Vollmer. 2006. Perspectives for cryptographic long-term security. *Commun. ACM* 49, 9 (2006), 50–55.
[6] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security Symposium 2018*. USENIX Association, 991–1008.
[7] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. 2018. SgxPectre attacks: Leaking enclave secrets via speculative execution. *CoRR* abs/1802.09085 (2018). arXiv:1802.09085 http://arxiv.org/abs/1802.09085
[8] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. 1985. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *Foundations of Computer Science (FOCS'85)*. 383–395.
[9] Victor Costan, Ilia A Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal hardware extensions for strong software isolation. In *USENIX Security Symposium 2016*. USENIX Association, 857–874.
[10] D-Wave. 2015. Announcing the D-Wave 2x quantum computer. https://www.dwavesys.com/blog/2015/08/announcing-d-wave-2x-quantum-computer/
[11] Michael D. Garris, James L. Blue, Gerald T. Candela, Patrick J. Grother, Stanley Janet, and Charles L. Wilson. 1997. NIST form-based handprint recognition system (release 2.0). *Interagency/Internal Report (NISTIR) - 5959* (1997).
[12] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. 1995. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology – CRYPTO'95*. Springer, 339–352.
[13] Intel. 2014. Software Guard Extensions Programming Reference, Revision 2.
[14] Per Larsen, Andrei Homescu, Stefan Brunthaler, and Michael Franz. 2014. SoK: Automated software diversity. In *IEEE Symposium on Security and Privacy (S&P'14)*. IEEE, 276–291.
[15] Torben Pryds Pedersen. 1992. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO'91*. Springer, 129–140.
[16] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
[17] Chia-che Tsai, Donald E. Porter, and Mona Vij. 2017. Graphene-SGX: A practical library OS for unmodified applications on SGX. In *USENIX Annual Technical Conference (USENIX ATC)*. USENIX, 645–658.
[18] T. F. Watson, S. G. J. Philips, Erika Kawakami, D. R. Ward, Pasquale Scarlino, Menno Veldhorst, D. E. Savage, M. G. Lagally, Mark Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen. 2018. A programmable two-qubit quantum processor in silicon. *Nature* 555, 7698 (2018), 633.