Practical and Verifiable Electronic Sortition

Hsun Lee
Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
leexun@csie.ntu.edu.tw

Hsu-Chun Hsiao
Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
hchsiao@csie.ntu.edu.tw

***Abstract*—Existing verifiable e-sortition systems are impractical due to computationally expensive verification (linear to the duration of the registration phase, T) or the ease of being denial of service. Based on the advance in verifiable delay functions, we propose a verifiable e-sortition scheme whose result can be efficiently verified in constant time with respect to T. We present the preliminary design and implementation, and explore future directions to further enhance practicability.**

***Index Terms*—verifiable sortition, verifiable delay functions**

I. Introduction

Electronic sortition (e-sortition) plays a principal role in democratic societies. It enables fair distribution of limited resources, such as the right to rent social housing [2] or to pre-order masks [7] [9] during epidemic prevention. A typical e-sortition system randomly selects a subset from the registered users by using a centralized server. To reduce trust in the centralized server, it is desirable to have *verifiable e-sortition*, whose fairness can be publicly verified.

However, existing verifiable e-sortition schemes are impractical due to computationally expensive verification or the ease of being denial of service. For example, some systems [3, 5] apply *delay functions*, a kind of moderately hard cryptographic functions, to prevent malicious users from manipulating the result before the end of the sortition registration. These systems require verifiers to re-compute the delay function, which takes longer than the time of the registration phase (e.g.,several days). Some systems [4, 8] rely on commit-reveal protocols to prevent such manipulation, but they are vulnerable to denial of service attacks caused by participants withholding the confirmation messages.

In order to construct a practical verifiable e-sortition system, we adopt the *verifiable delay function (VDF)*, which is a special type of delay functions whose output can be efficiently verified [1]. Our scheme can securely generate unpredictable and unbiased pseudo-random result without any trusted third party, and improve the time complexity of delay function verification from linear to constant.

A. The scheme

Throughout this paper, x_u denotes a string generated by a user u, y denotes the verifiable random output (which will seed a public pseudo-random number generator for e-sortition), and π denotes the proof of the output correctness. T is a public parameter indicating the computation time, which must be slightly longer than the time range of the registration. N is the final number of registered users. Furthermore, our scheme can be initialized without any trusted third party, because we apply the implementation of VDF proposed by Wesolowski [10]. Our scheme can be divided into three phases:

1) **Registration.** Each user u selects and sends a arbitrary x_u to the server. The server receives x_u and returns a digitally signed confirmation as a proof of registration.
2) **Result generation.** The server builds a Merkle tree with all x_us as the leaves, and obtains the root of the Merkle tree, x_{root}. The server then evaluates the VDF function $VDFEval(x_{root}, T) = (y, \pi)$. After computation, the server seeds a public PRNG using y to select the winners. Finally, the server publishes (y, π) to all users.
3) **Verification.** Every user u can verify whether his or her x_u was correctly included during result generation. Specifically, the user requests the server for the Merkle audit path of x_u and re-computes the root hash of the Merkle tree to verify the inclusion proof. After completion, the user computes $VDFVerify(x_{root}, y, \pi, T) = \{Yes, No\}$ to determine the correctness of (y, π). Notice that $VDFVerify$ is much faster than $VDFEval$, because it is not a re-computation of $VDFEval$. If any of the verification fails, the user can prove the sortition is invalid by providing the information above.

II. Result

Compared with prior work, our construction reduces the time and memory complexity of the verification phase by applying the VDF and Merkle tree (Table I). In addition, we decrease the maximum time of H_{prime} function in VDF verification in web browsers from 10.7s to 683ms by skipping unnecessary procedures of primality tests (Appendix A).

TABLE I
Complexity Improvement

	Verification of delay function	***Size of inclusion proof***
[1]	$\mathcal{O}(T)$	$\mathcal{O}(N)$
[2]	$\mathcal{O}(T)$	$\mathcal{O}(\log(N))$
Our work	$\mathcal{O}(1)$	$\mathcal{O}(\log(N))$

For future work, we are extending our scheme to provide a verifiable e-sortition service that abstracts the implementation of VDFs. We envision that such a loosely coupled service can allow existing non-verifiable e-sortition systems to rapidly transit to verifiable ones.

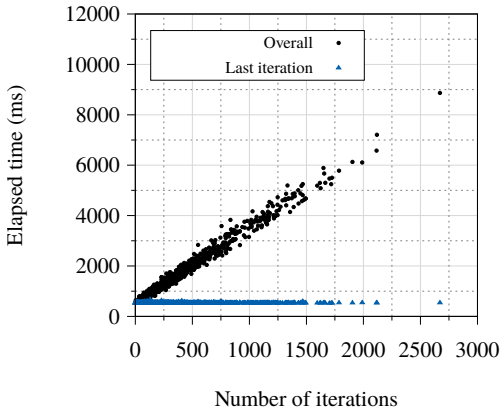

Fig. 1. H_{prime} in Safari 12.1 on MacBook Pro with 2.7 GHz Intel Core i5

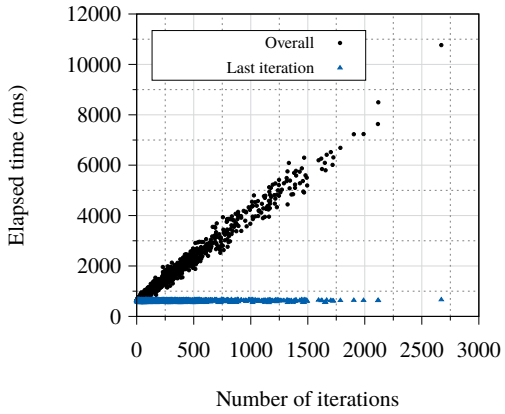

Fig. 2. H_{prime} in Safari 12.1 on iPhone 6 with Dual-core 1.4 GHz Typhoon

Acknowledgements. This research was supported by the Ministry of Science and Technology of Taiwan under grant MOST 109-2636-E-002-021.

References

[1] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable delay functions". In: *Annual international cryptology conference*. Springer. 2018, pp. 757–788.

[2] Chin-Oh Chang and Shu-Mei Yuan. "Public housing policy in Taiwan". In: *The Future of Public Housing*. Springer, 2013, pp. 85–101.

[3] Sherman SM Chow, Lucas CK Hui, Siu-Ming Yiu, and KP Chow. "Practical electronic lotteries with offline TTP". In: *Computer Communications* 29.15 (2006), pp. 2830–2840.

[4] Biao He and Yu Wei. "Electronic sortition". In: *Proceedings. The 2009 International Symposium on Intelligent Information Systems and Applications (IISA 2009)*. Citeseer. 2009, p. 203.

[5] Yi-Ning Liu, He-Guo Liu, Lei Hu, and Jin-Bing Tian. "A New Efficient E-Lottery Scheme Using Multi-Level Hash Chain". In: *2006 International Conference on Communication Technology*. IEEE. 2006, pp. 1–4.

[6] Chai Network. 2020. [Online]. Available: https://github.com/Chia-Network/chiavdf. (accessed June 15, 2020).

[7] eMask Ordering System. 2020. [Online]. Available: https://emask.taiwan.gov.tw/msk/index.jsp. (accessed May 11, 2020).

[8] Rasoul Ramezanian and Mohsen Pourpouneh. "Decentralized Online Sortition Protocol." In: *ISeCure* 10.1 (2018).

[9] C. Jason Wang, Chun Y. Ng, and Robert H. Brook. "Response to COVID-19 in Taiwan: Big Data Analytics, New Technology, and Proactive Testing". In: *JAMA* 323.14 (Apr. 2020), pp. 1341–1342.

[10] Benjamin Wesolowski. "Efficient verifiable delay functions". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 379–407.

Appendix A
Experiment Result

We conduct preliminary experiments to evaluate the verification speed in web browsers on laptop (Fig. 1) and mobile phone (Fig. 2). We use the VDF implemented by Chai Network [6], because Chia opened a competition for the fastest VDF to precisely evaluate the security of the Time-lock assumption in VDF [10]. Additionally, we modify the implementation of the VDF to cross-compile it from C++ into WebAssembly to ensure future portability.

However, we expect there will be a performance downgrade after moving native program into browser. For example, the procedure of primality test. The VDF proposed by Wesolowski [10] requires the server and the users themselves to compute the x_{root} into a negative prime d of large absolute value, and such that $d \equiv 1 \pmod 4$. This procedure involves $H_{prime}(s) = p$, a deterministic algorithm takes arbitrary string s, iteratively performs hash and primality test, and finally returns a large psuedoprime p ($2^{1023} \leq p < 2^{1024}$). It might take lots of iterations to complete and thus increases potential overhead.

To estimate the overhead of primality test, we generate 1024 sample strings as s to measure the elapsed time of H_{prime} in web browsers. We use *mpz_probab_prime_p* in GMP 6.2.0 with 50 repetitions to perform the test. As shown in Fig. 1 and 2, it takes around 547ms to pass the final primality test, which occurs in every final iteration before H_{prime} returns. Namely, most of time in H_{prime} is spent on those failed primality tests.

We propose a solution to reduce this overhead. The server can publish a value i, the number of the iterations to find the psuedoprime, with (y, π) after result generation. Therefore, users can perform $i - 1$ iterations and only do the primality test in the ith iteration to ensure the final d is truly a psuedoprime. By this method, the maximum time of H_{prime} in our experiment on iPhone 6 can be decreased from 10.7s to 683ms. The data of these experiments can be found at https://github.com/leexun/verifiable-sortition-system.