Mike Sadowski

CP682B-OC1

Thursday, April 8th 2021

Public-Key Generation with Verifiable Randomness: Paper Review

"One thing that traditional computer systems aren't good at is coin flipping," says Steve Ward, Professor of Computer Science and Engineering at MIT's Computer Science and Artificial Intelligence Laboratory (Rubin, J. M, 2011). Computers are deterministic, if you give them an input, they are going to produce an output. This output will be the same every time, resulting in a lack of randomness. "If you go to an online poker site, for example, and you know the algorithm and seed, you can write a program that will predict the cards that are going to be dealt." Truly random numbers make such reverse engineering impossible (Rubin, J. M, 2011).

Similar to online poker games, encryption algorithms need truly random numbers in order to keep what they are encrypting a secret. However, generating truly random numbers is a very expensive process. Verification is also an important aspect as we need a way to verify that these numbers cannot be reproduced with a specific input in order to ensure the security of the encryption. This is especially the case with public key cryptography as the key used to encrypt messages for the recipient to decrypt are made public. Adversaries are free to inspect the key and try to recover the private key at their will.

An example of such a weakness in public key cryptography is the famous ROCA vulnerability. The ROCA vulnerability is a "cryptographic weakness that allows the private key of a key pair to be recovered from the public key in keys generated by devices with the vulnerability" (Nemec M et al, 2017). The flaw in these systems is the generation of the numbers. "The flawed key-generation algorithm selects specific prime numbers as part of the private key instead of generating uniformly random primes and many certified devices were shown vulnerable (e.g., Estonian and Slovakian smartcards and standard cryptographic libraries)" (Nemec M et al, 2017). This raises a concern as a user's information is not truly safe and these compromised encryption algorithms cannot guarantee security. If a random prime number was used in the key generation, this flaw would simply not exist.

Authors Olivier Blazy, Patrick Towa, Damien Vergnaud proposed a fantastic system for verifying the randomness in algorithms. In their paper titled "Public-Key Generation with Verifiable Randomness", they propose a protocol for all key generation algorithms based on probabilistic circuits to prove their security. "We give a generic protocol for all key-generation algorithms based on probabilistic circuits and prove its security. We also propose a new protocol for factoring-based cryptography that we prove secure in the aforementioned model. This relies on a new efficient zero-knowledge argument for the double discrete logarithm problem that achieves an exponential improvement in communication complexity compared to the state of the art, and is of independent interest" (Olivier Blazy et al, 2020).

Inspired by the Bellare-Pointcheval-Rogaway model for authenticated key exchange, they "propose a game-based model that covers con- current protocol executions with different instances of protocol algorithms" (Olivier Blazy et al, 2020). The communication between the user and the CA (certificate authority) is assumed to be carried over an insecure, asynchronous channel (in which adversaries are able to tap and modify messages between user and CA). The adversary is split into two algorithms: (1) the sampler which provides the randomness sources to the user and the CA (for multiple instances of the protocol) and (2) the distinguisher which tries to gain information from the generated public key. The protocol is deemed secure if the distinguisher is unable to do so assuming that the entropy of either random source is high enough (Olivier Blazy et al, 2020).

Their aim with this protocol is to provide a measurable level of security by examining the randomness of the numbers used in key generation. They maintain user privacy by enforcing the use of random numbers so that the adversary has no additional information on the key that was generated using verifiable randomness. By improving the randomness quality, the adversary (other than the CA) "has no additional information on the secret key compared to a key generated by the real key-generation algorithm on uniform randomness" (Olivier Blazy et al, 2020). The resulting public key that was generated is not leaking any information whatsoever. In particular, a faulty user algorithm cannot use it to convey any information. In this sense, the CA certifies to the end user that she can securely use the generated key (Olivier Blazy et al, 2020).

Previous work done by Ari Juels and Jorge Guajardo tried to accomplish the same goals as the protocol discussed in this paper. In their paper titled "RSA Key Generation with Verifiable Randomness" published in 2002, the authors proposed a formal security model for verifiable random key generation. However, their model was unfortunately not powerful enough to stop all real-world threats and fell short of what the authors were aiming to accomplish with their model. First off, the model was restricted to only public-key cryptosystems "where a given public key corresponds to a unique secret key" (Air Juels et al, 2002). It only considers standalone or independent key generation. Meaning, it does not prevent attacks where several public keys are generated with correlated random sources (Olivier Blazy et al, 2020). In addition to these reasons, it also only "bounds the distance that a dishonest user algorithm can generate a given key to that of an honest algorithm executing the key generation protocol" (Olivier Blazy et al, 2020).

The model proposed by Olivier Blazy, Patrick Towa, Damien Vergnaud does not suffer from the shortcomings of the model proposed from Ari Juels and Jorge Guajardo as it allows for multiple runs of the protocol and captures the resistance to exfiltration of information (with only the narrow-band subliminal channel from the "halting attack") and it guarantees security with concurrent sessions (and is thus much stronger than security considered in cryptographic reverse firewalls) (Olivier Blazy et al, 2020).

While no cryptosystem is perfect there are some that are definitely better than others. With the help and use of verifiable randomness within these systems, a second layer of protection can be added as the algorithms used to produce the keys can in some aspects be quite deterministic. This is dangerous as potential attackers have the potential to determine private keys and gain access to user information. The proposed protocol/model by the authors of this paper did a great job providing a measurable way to determine the randomness of numbers used within public key generation.

Works Cited

1. Rubin, J. M. (2011, November 1). Can a computer generate a truly random number? MIT Engineering. https://engineering.mit.edu/engage/ask-an-engineer/can-a-computer-generate-a-truly-random-number/
2. Nemec, M., Sýs, M., Svenda, P., Klinec, D., Matyas, V.: The return of copper- smith's attack: Practical factorization of widely used RSA moduli. In: Thuraising- ham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1631–1648. ACM Press (Oct / Nov 2017)
3. Juels, A., Guajardo, J.: RSA key generation with verifiable randomness. In: Nac- cache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 357–374. Springer, Heidelberg (Feb 2002)