Tamper and Leakage Resilient von Neumann Architectures from Continuous Non-Malleable Codes

An Invited Talk

Jesper Buus Nielsen

Department of Computer Science
Aarhus University

Abstract. We present the notion of continuous non-malleable codes along with an instantiation and we show how to use them to securely compute any keyed cryptographic primitive on a computational architecture with a single constant size untamperable CPU and a tamperable and leaky memory in which both the secret key and the program of the cryptographic primitive is located.

1 Introduction

The notion of non-malleable codes introduced by Dziembowski, Pietrzak and Wichs in [2] is a relaxation of the notions of error correcting and error detecting codes. Informally, a code is non-malleable if an adversary trying to tamper with an encoding of a given message can only leave it unchanged or modify it to the encoding of a completely unrelated value.

Continuous non-malleable codes (CNMC) is an extension of the standard non-malleability security notion, where we allow the adversary to tamper with an encoding several times and use its knowledge of the observed effects of the tampering in the subsequent tampering attacks. This is in contrast to the standard notion of non-malleable codes where the adversary is only allowed to tamper a single time with an encoding.

The notion of continuous non-malleable codes was introduced by Faust, Mukherjee, Nielsen and Venturi in [3] where also the first such code was constructed. We present the code from [3]. The code is based on the inner-product function, collision-resistant hashing and non-interactive zero-knowledge proofs of knowledge. We also touch on later work by Coretti, Maurer, Tackmann and Venturi[1, 5] and Jafargholi and Wichs[5] constructing information-theoretically secure continuous non-malleable codes.

We then present how to use continuous non-malleable codes to protect arbitrary cryptographic primitives against tampering attacks. Previous applications of non-malleable codes to this problem required to perfectly erase the entire memory of the computational architecture after each execution. This of course makes it impossible to have the program of the primitive sit in the memory of

the computational architecture and forces the program to be hardcoded into the architecture itself, forcing essentially a circuit model of computation. In practice this would mean producing a specialised piece of hardware for each primitive that we would like to compute in a tamper-resilient manner. Continuous non-malleable codes were introduced exactly to avoid this limitation.

We present the tamper and leakage-resilient random-access memory architecture from [4]. The architecture has one CPU that accesses a memory. The memory is subject to leakage and tampering. So is the bus connecting the CPU to the memory. We assume that the computation of the CPU is leakage and tamper free. For a fixed value of the security parameter, the CPU has constant size. Furthermore, the design of the CPU is completely independent of the program to be run and its internal registers are non-persistent, i.e., all secret registers are reset between invocations. The most prominent consequence of having a constant size CPU with no persistent secret memory is that the code of the program must be stored in the memory and so must all secrets. Therefore both program and secrets will be subject to tampering and leakage.

We construct a compiler for this architecture which transforms any keyed cryptographic primitive into a program where the key is encoded and stored in the memory along with the program to evaluate the primitive on that key. This result reduces the problem of shielding arbitrarily complex computations to protecting a single, constant-size component. The compiler only assumes the existence of a continuous non-malleable code and is therefore information-theoretically secure if based on an information-theoretically secure continuous non-malleable code.

References

1. S. Coretti, U. Maurer, B. Tackmann, and D. Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In Y. Dodis and J. B. Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2014. Proceedings*, Lecture Notes in Computer Science. Springer, 2015.
2. S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. In A. C. Yao, editor, *Innovations in Computer Science - ICS 2010. Proceedings*, pages 434–452. Tsinghua University Press, 2010.
3. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. Continuous non-malleable codes. In Y. Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488. Springer, 2014.
4. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. Leakage-resilient signatures with graceful degradation. In J. Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th International Conference on Practice and Theory in Public-Key Cryptography. Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 579–603. Springer, 2015.
5. Z. Jafargholi and D. Wichs. Tamper detection and continuous non-malleable codes. In Y. Dodis and J. B. Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2014. Proceedings*, Lecture Notes in Computer Science. Springer, 2015.