Cryptography with Disposable Backdoors

Kai-Min Chung[1], Marios Georgiou[2], Ching-Yi Lai[3] and Vassilis Zikas[4]

[1]Academia Sinica, kmchung@iis.sinica.edu.tw
[2]City University of New York, mgeorgiou@gradcenter.cuny.edu
[3]Institute of Communications Engineering, National Chiao Tung University, cylai@nctu.edu.tw
[4]University of Edinburgh, vzikas@inf.ed.ac.uk

June 16, 2019

Abstract

Backdooring cryptographic algorithms is an indisputable taboo in the cryptographic literature for a good reason: however noble the intentions, backdoors might fall in the wrong hands, in which case security is completely compromised. Nonetheless, more and more legislative pressure is being produced to enforce the use of such backdoors.

In this work we introduce the concept of *disposable cryptographic backdoors* which can be used only once and become useless after that. These exotic primitives are impossible in the classical digital world without stateful and secure trusted hardware support, but, as we show, are feasible assuming quantum computation and access to classical stateless hardware tokens.

Concretely, we construct a disposable (single-use) version of message authentication codes, and use them to derive a black-box construction of stateful hardware tokens in the above setting with quantum computation and classical stateless hardware tokens. This can be viewed as a generic transformation from stateful to stateless tokens and enables, among other things, one-time programs and memories. This is to our knowledge the first provably secure construction of such primitives from stateless tokens.

As an application of disposable cryptographic backdoors we use our constructed primitive above to propose a middle-ground solution to the recent legislative push to backdoor cryptography: the conflict between Apple and FBI. We show that it is possible for Apple to create a one-time backdoor which unlocks *any single* device, and not even Apple can use it to unlock more than one, i.e., the backdoor becomes useless after it is used. We further describe how to use our ideas to derive a version of CCA-secure public key encryption, which is accompanied with a disposable (i.e, single-use, as in the above scenario) backdoor.

1 Introduction

The use of strong cryptographic primitives for widely available devices has led to controversial debates between the computer security community and public policy makers. On the one hand, law enforcement agencies argue that allowing access to such primitives enables cyber-terrorists to use it to elude detection, and thereby reduces the effectiveness of law enforcement. On the other hand, the computer security—and most vocally the cryptographic—community argues that allowing everyone to use such strong primitives can help protect their security and make cybercrime less effective in the first place.

In this work we put forth the concept of *disposable-backdoor cryptography.* Intuitively, given a keyed cryptographic primitive, e.g., authentication/identification or encryption, we can define its "disposable-backdoor" version. In this version, the key-generation algorithm outputs, additionally, a quantum key/backdoor that can be used to gain knowledge about the secret information, e.g., the plaintext included in a given ciphertext. In a nutshell security ensures the following: (1) in absence of the above additional quantum key the scheme achieves the same guarantees as its non-backdoored counterpart; and (2) anyone given access to the backdoor can use it only once. (E.g., an adversary participating in two CCA games and given access to the backdoor might only win in one of these games.)

In theory, the above transformation is feasible by using the so called *one-time programs* [22]. Informally, these are programs that can be executed once and then become useless, i.e., they terminate and cannot be reused. However, this exotic primitive is known to be impossible both in the classical world [22] and in the quantum world [10] without any further setup assumptions. We overcome this limitation by defining one-time programs relative to a *stateless classical* oracle—this corresponds to equipping the programs with access to a *classical* honestly generated *stateless* hardware token. We prove that one-time programs are possible in this stateless (classical) hardware token model. This is, to our knowledge, the first construction of one-time programs (consequently, also of one-time memories) from stateless (classical) tokens. We note in passing that such a construction is impossible in the classical setting, i.e., without quantum computation (cf. Section 1.2).

Our construction of one-time programs from stateless (classical) tokens is a special instantiation of a more general transformation—which we provide—that uses quantum computation to reduce a stateful (classical) oracle to a stateless (also classical) one. We view the fact that quantum cryptography enables such a transformation, and its implications, as a new advocate of the relavance of quantum cryptography for applications, beyond the standard application of quantum key distribution [6]. Furthermore, this transformation is a considerable step in the long line of research investigating the power of stateless tokens, which are known to be a strictly weaker assumption than their stateful counterparts. For example, one-time programs are trivial in the stateful token model—simply run the program inside a token that is instructed to halt after the first use—whereas they can be easily shown impossible from stateless tokens without quantum computation—the standard argument that "if I can run a classical stateless program on any one input I can also run it on any two inputs" [22] trivially applies here. Furthermore, stateful tokens are in general susceptible to resetting attacks. Using the stateful-to-stateless transformation we can neutralize such attacks in the quantum world.

Our results, and in particular the above transformation, have several applications. Since our techniques enable replacing stateful (classical) hardware tokens by stateless ones, they can be applied in a variety of settings where stateful tokens are necessary, e.g., *physical (multi-party) computation* [16, 17]. Furthermore, cryptography with disposable backdoors can be used when one might be willing to selectively compromize security of some sessions but does not want whoever has the power to do so to be able to use this powers indefinitely. For example, our techniques allow for a middle-ground solution to one of the most popular instances of the above debate between security and policy makers, namely, the case of the FBI against Apple Inc. [26]. In short, the FBI wanted Apple Inc. to create and electronically sign a new software that would enable the FBI to unlock a work-issued iPhone 5C, recovered from one of the shooters in a December 2015 terrorist attack in San Bernardino, CA [30]. Apple Inc. refused to comply with this request. The main argument was that such a software would effectively serve as a backdoor and anyone who got his hands on it would be able to breach the privacy of the smartphone's holder at will. Thus if the backdoor fell in malicious hands, it would yield unprecedented havoc. Indeed, unlike what was suggested by the FBI [11], there does not seem to be a way to create some digital information that can only be used by the "good guys" and becomes useless (or less functional) in the hands of malicious actors that manage to steal the information the good guys hold.

Using our techniques we can prove that with the help of quantum storage, we can develop backdoors that can be used *only once* to bypass the security of any one device from a defined set, and then become completely useless. In particular, we show that these single-use backdoors allow an arbitrary smartphone to be unlocked and they become useless after this phone has been unlocked. Most importantly, not even Apple itself is able to unlock more than one phones, which ensures that anyone getting access to Apple's information can at most break into one phone.

More concretely, we demonstrate a construction of "disposable backdoored devices" as a way to resolve the smartphone vs. law enforcement conundrum. Concretely, our construction allows a device (e.g., smartphone) vendor to embed in its devices a stateless content locking mechanism, and create (and locally store on the vendor's side) a disposable unlocking backdoor that can unlock *exactly one* smartphone—any one from a specified set. (We stress that in our construction the smartphone is a standard classical device, but the security of our scheme would hold even if it were a quantum device.) To make our scheme most general we look at the question of how we can dynamically extend the set of devices/smartphones that can be unlocked, even after the generation of the original backdoor. An obvious solution would be to create a new one-time backdoor and update every phone in the set we want it to be able to unlock. However, this solution is clearly not scalable, as it requires such an update of *all* existing devices—old and new—every time a new (batch of) devices needs to be released. Instead, we provide a mechanism for extending the set of devices that can be unlocked with the existing single-use backdoors without interacting with the ones already in the set. This mechanism relies on a black-box technique for extending one-time memories, which might be of independent interest.

We note in passing that we acknowledge that backdooring of any type is a charged topic in the cryptographic and security literature. Nonetheless, we view our work as an investigation of a theoretically interesting middle-ground, which might allow the cryptographic community to be prepared against public pressure that might demand legislative instatement of such backdooring.[1] Indeed, one might argue that given such a functionality, law enforcement would have the ability to surveil anyone they choose. Although this is in-principle true, if it is publicly known that there is only one (or a small limited number of)[2] such disposable backdoors, law enforcement will have to use them with care to make sure that they do not become unavailable in high-profile cases where the public might agree that their use is acceptable.

1.1 Overview of our techniques

The new cryptographic tool for our constructions, which we define here, is *disposable message authentication codes (DMAC)*. This primitive allows someone having a quantum key to use it *exactly once* in order to compute an authentication tag to any message. In fact, its construction is closely related to that of quantum money [3].

We use DMACs to devise a generic reduction of stateful to stateless *classical* oracles. These oracles correspond to the notion of stateless and honestly-generated (classical) hardware tokens. Concretely, for any given stateful token T we show how to generate code for a stateless token along with one-time (quantum) backdoors, so that we can use them to devise a protocol that implements the stateful token T with unconditional security. Importantly, the tokens/oracles considered in this work can only be queried in a *classical* manner. In fact, such a reduction would become impossible if quantum superposition queries to the stateless token are allowed (see further discussion below), and restriction to classical queries is the key to enable our transformation. We believe that such restriction is a mild assumption that holds in most existing instantiations of primitives, such as smartcards, trusted co-processors, etc.

The idea behind our stateful-to-stateless transformation is to generate a (quantum) DMAC tagging key to be used to emulate each round of interaction with the stateful token/oracle T. Denote this key sequence by $k_1, \dots, k_m$. We then derive (a program for) our stateless oracle which given any sequence of inputs $(x_1, t_1), \dots, (x_q, t_q)$, checks that each t_i is a valid DMAC tag corresponding to the i-th instance, and if this is the case, perform the same computation that T would on input $x_1, \dots, x_q$. The security of the DMAC will ensure that none can receive responses from the stateless tokens on query-sequences that have different prefixes. Hence, once any sequence of q queries has been successfully submitted, there is no way to "rewind" the token and query it on a different sequence, which emulates the behavior of the stateful token T.

Having built such a stateful-to-stateless transformation, we go on to create our transformation from any program to its one-time version relative to a classical oracle. The idea here is to first describe the one-time program as a stateful oracle/token and then use the above transformation to turn it into a stateless one. Finally, using one-time programs and one-time memories as their special case, we address the problem of building one-time backdoors for unlocking devices. To achieve this, we allow the vendor to create phones that are hardcoded with a fresh symmetric-encryption key. When a phone locks, it uses this key to encrypt its state and subsequently discards the key. Now, the key can only be accessed through the one-time memory.

1.2 Related Literature

Our work combines elements from several different areas of classical and quantum cryptography, ranging from quantum money to quantum tokens for message authentication codes and one time memories.

DMACs can be thought as the symmetric key version of the one-time tokens for digital signatures [12]. However, as symmetric-key primitives, DMACs can be—and in our work are—implemented by information-theoretic constructions, which tolerate unbounded adversaries. Other related primitives are secret-key quantum money, e.g., [2, 3], and *quantum retrieval games* (QRGs) [18, 29]. A QRG differs from a DMAC in

[1] Such cryptography-restrcting legislation has been used in the past and several countries are starting to reinstate it [15].

[2] Our results can be extended to an a-priori limited-use backdoor version, cf. Remark 1 in Appendix B.

that it does not allow the adversary to have access to a verification oracle. Gavinsky [18] proved that such a primitive is enough to construct secret-key quantum money by having a 3-round verification protocol with the bank. Later Georgiou and Kerenidis [19] improved this construction by achieving only one round verification.

One-time memories (OTMs) were first proposed by Bennett et al. [7], under the term *quantum multiplexing*. They are devices that contain two secrets but only one of the two can be extracted. Goldwasser et al. [22] proved that OTMs are enough to achieve one-time programs, i.e. programs that can be run only once. Broadbent et al. [10] extended this result to the quantum setting by showing that quantum OTMs are enough to construct quantum one-time programs. Despite being very close to a non-interactive version of oblivious transfer, OTMs are impossible to achieve in the plain model, both in the classical world and in the quantum world, even in the computational setting. The classical impossibility comes from the fact that a memory should correspond to a classical bitstring and therefore by copying the bitstring we can easily extract both secrets.

Quantumly, the no-cloning theorem seems to enable a construction of OTMs. Unfortunately, this is still impossible, since it is theoretically possible to extract a value from a quantum state with probability close to 1 without collapsing the state. Thus we can invert the extraction procedure of the first value and then extract the second value as well [1]. This impossibility underlines the necessity of additional setups, e.g., classical tokens, to achieve one-time primitives. For details see Winter's "Gentle Measurement Lemma" [31] which was later improved by Ogawa and Nagaoka [28] as well as Aaronson's "Almost as good as new lemma" [1]. The lemmata state informally that a post-measurement state of an almost-sure measurement will remain close to its original.

The idea of using quantum information to reduce stateful to stateless tokens was originally posted by Broadbent, Gharibian, and Zhou [8], but in a model different from ours. [8] gave a candidate prepare-and-measure scheme, but the original security proof was incomplete. Later, in a concurrent work, Broadbent et al. [9] showed a prepare-and-measure scheme that is secure against an adversary making a linear number of queries. To our knowledge, our work is the first to provide a provably secure reduction of (classical) stateful to stateless tokens in the quantum setting that is secure against an adversary making a polynomial number of queries (though the results are not directly comparable given the difference in the models).

Finally, starting with the work of Katz [25], several works investigated the sufficiency of stateless hardware tokens—not necessarily honestly generated—for secure (multi-party) computation, e.g., [14, 24]. These results are in the classical setting and, therefore, do not imply feasibility of one-time programs or OTMs, or a generic reduction of stateful to stateless tokens.

1.3 Organization of the Paper

The remainder of the paper is organized as follows. In Section 2, we provide some preliminaries, notation, and lay down the model of computation. In Section 3, we provide our definitions and instantiation of disposable MACs (DMACs). Then, in Section 4, we provide our reduction of stateful (classical) tokens to stateless (classical) tokens using DMACs. In Section 5, we provide our definition of one-time memories (OTMs) relative to a (classical) oracle, which captures the security of OTMs in the stateless hardware token model. Finally, in Section 6, we present our constructions of one-time (disposable) backdoor devices—including the definition and construction of extendible OTMs. Due to space limitation, certain primitives and proofs have been moved to the clearly marked appendix which is referred to appropriately. At the end of the appendix, we also include a direct construction of One-time Programs (Section C) along with our definition and construction of CCA-secure encryption with disposable backdoors (Section D).

2 The model

In this section we describe our model of computation. Before that, we provide some necessary terminology and notation: A function f is negligible if $f(n) \in o(1/\mathsf{poly}(n))$ for any polynomial poly. For two quantum states ρ, σ we denote by $\Delta(\rho, \sigma)$ their trace distance $\frac{1}{2}||\rho - \sigma||_1$. If the trace distance between two quantum states is negligible then we will denote this by $\rho \sim_s \sigma$ and we will say that the quantum states are statistically indistinguishable. For two quantum algorithms A, B that possibly have inputs and oracle access to some algorithms, we will write $A \sim_s B$ if the quantum states that they output are statistically indistinguishable.

Our results consider systems that might be classical and/or quantum computation enabled. E.g., our disposable-backdoor device application considers a classical smartphone but the vendor—who also stores the one-time backdoor—can perform quantum computation, i.e., create, store, and measure qubits. Quantum-enabled parties can store quantum states and exchange them with each other. We consider information-theoretic security in the quantum setting, i.e., the adversary is a computationally unbounded quantum machine, but can make only a polynomial number of (classical) queries to the stateless token—this polynomial, however, is not known to our constructions.

Our constructions are assumed access to *classical* tokens which allow only classical access—in particular, the only way to interact with such a token is to hand it as input a classical string. We devise natural definitions of the primitives we construct as oracle algorithms in the plain model of computation (cf. Section 5.) More concretely, we model algorithms with access to classical tokens as oracle algorithms, where the oracle is classical and offers the same functionality as the corresponding token. In particular, our classical oracles can only be queried with classical strings (no quantum interfaces), may only perform classical computations, and produce classical output. Throughout this work we use the terms oracle and token interchangeably. Recall that the assumption of classical-only tokens is not just consistent with the capabilities of common devices that can be used as hardware tokens, e.g., smartcards, but it is also minimal since quantum-accessible tokes are known to be insufficient to circumvent the impossibility of one-time primitives [1].

3 Disposable MACs

In this section we introduce the notion of *disposable message authentication codes* (DMACs, in short) and demonstrate how they can be implemented. In a nutshell, DMACs are a one-time version of classical MACs, i.e., the secret key can be used to authenticate only a single message and then becomes useless (except for verification purposes). We remark that DMACs are different from what is called one-time MACs in the cryptographic literature. Indeed, the latter are MACs that preserve their security as long as they are used at most once, i.e., they could be used for tagging more than one message but this would render them insecure/forgeable. Instead, DMACs do not allow anyone—honest or adversarial—to use the same MAC key to tag two different messages.[3]

Concretely, classical message authentication codes (MACs) are symmetric-key primitives that allow two parties, who share a key, to exchange messages in an authenticated manner. In a nutshell, any of the parties can use the key within a tagging algorithm Tag to create an authentication tag t to any given message (t is often referred to as a *MAC tag*). The security of the scheme ensures that only the message/tag pairs generated with the shared key will always be accepted by the receiver (completeness); however, no adversary who does not know the key can forge an acceptable authentication tag on a new message (existential unforgeability).

DMACs are MACs whose key can be used to tag exactly one message. This is achieved by adding a quantum state as a part of the tag-generation key. This quantum state allows whoever holds it to tag any one message of their choice. We remark that DMACs authenticate classical (not quantum) messages. The formal definition follows.

Definition 1 (Disposable single-bit MACs). *A single-bit disposable MAC (DMAC) is a triplet of algorithms* $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ *defined as follows:*

- $\mathsf{Gen}(1^n) \rightarrow (s, \rho)$ *is a quantum algorithm that takes as input a security parameter n and returns a disposable (secret) key-pair consisting of a classical bit-string s of size n and a quantum state ρ. We will refer to ρ as the disposable (part of the) key.*
- $\mathsf{Tag}(\rho, b) \rightarrow t$ *is a quantum algorithm that takes as input a quantum state ρ and a bit b, and returns a classical tag t.*

[3] There is an unfortunate clash in terminology in the literature as one-time programs and one-time memories achieve a similar "one-timeness" as DMACs, which is different from what one-time MACs and one-time signatures achieve. Here, we choose to use the term disposable for MACs to avoid ambiguity.

- $\mathsf{Ver}(s, b, t) \to \{0, 1\}$ *is a classical algorithm that takes as input a secret s, a bit b, and a tag t, and either accepts or rejects.*

The security of DMACs is similar to the security of the original MACs (we refer to [21] for a formal definition), but instead of existential unforgeability, it requires *disposable existential unforgeability* property which forbids the adversary from creating valid tags for two different (classical) messages with a single tagging key. Note that unlike standard EUCMA-security, the adversary is not given access to a MAC-tag generation oracle—since the disposable key can be used only once. However, we do allow the adversary to use a classical verification oracle that, given a received (message,tag)-pair (b, t), responds whether or not $\mathsf{Ver}(s, b, t) = 1$.

Definition 2 (Security of DMACs)**.** *A DMAC (scheme)* $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ *is said to be secure if it satisfies the following properties:*

Completeness. *For any bit b and (s, ρ) output by* Gen*, it holds that* $\mathsf{Ver}(s, b, \mathsf{Tag}(\rho, b)) = 1$.

Disposable Existential Unforgeability (DEU). *Let V_s be a classical oracle, which on input a bit b and a tag t, outputs* $\mathsf{Ver}(s, b, t)$*. A DMAC* $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ *is DEU-secure if for any (computationally unbounded) quantum algorithm A with oracle access to V_s and polynomially many queries to V_s, it holds that* $\mathsf{Adv}_A^{DMAC} \leq \mathsf{negl}(n)$*, where*

$$\mathsf{Adv}_A^{DMAC} := \Pr_{\substack{(s,\rho) \overset{\$}{\leftarrow} \mathsf{Gen}(1^n) \\ (t_0,t_1) \overset{\$}{\leftarrow} A^{V_S(\cdot,\cdot)}(\rho)}} [\mathsf{Ver}(s, 0, t_0) = 1 \wedge \mathsf{Ver}(s, 1, t_1) = 1].$$

Note that in the above experiment, the adversary is not given the secret verification key s generated by Gen. This is the reason why this primitive is a secret-key primitive. In fact, in our constructions, if the adversary would get s, then he would be trivially able to generate valid MACs. This is because our constructions generate the disposable part of the key ρ from the secret s. It is also easy to verify that the DEU-security implies the classical notion of existential unforgeability [23] but without the MAC-tag generation oracle. Indeed, if the adversary had a process A for generating a valid MAC tag on a message without knowing any part of the key, then he could trivially break DEU-security by first running A to forge a MAC on one message b_0 and then use the disposable key ρ to generate a MAC tag for b_1 (the completeness property ensures that the latter will always succeed). In fact, intuitively, one should think of the quantum state as a one-time access to the tagging oracle.

Construction of DMACs. Despite our work being the first to provide a formal definition of DMACs, there are a couple of heavily related primitives studied in the quantum cryptography literature—e.g., *quantum retrieval games* [18], *unforgeable quantum tokens* [29], and one-time quantum digital signatures [12]. In fact as part of their one-time quantum digital signatures, Ben-David and Sattath [12] already developed the techniques and defined the algorithms that one needs for implementing DMACs. For completeness, we include this construction and the security argument in Appendix A. Notice that although we can create directly disposable MACs from disposable signatures, such an approach loses the information-theoretic security of the definition (since public-key signatures require computational assumptions). Instead, by being careful and using only part of [12], we achieve information theoretic security.

From single-bit to string DMACs. Definition 1 can be extended to the case where we want to tag a string of several (polynomially many) bits. In this case we require that there is no algorithm that can tag two different bit-strings. We refer to this primitive as *DMAC for strings* or *string DMAC*. The corresponding scheme and security definitions are trivially derived by modifying Definitions 1 and 2 so that instead of bits, they are applied to strings. For the remainder of this paper, we use DMAC to refer to string DMAC.

The construction of string DMACs from single-bit DMACs is straightforward: To generate tags for an n-bit string $m \in \{0, 1\}^n$, simply create n independent key-pairs $(s_1, \rho_1), \ldots, (s_n, \rho_n)$ for single-bit DMACs; the ith disposable key ρ_i is used to authenticate the i-th bit of m. The security intuition of the construction follows from the fact that since the key-pairs are honestly and independently generated, the single-bit

```
CODE FOR $A$ :
    $i \leftarrow 1$
    $state \leftarrow \perp$
    loop: On query $x$
        $(y, state) \leftarrow C_i(x, state)$
        $i++$
        return $y$
```

(a) Standard formulation

```
CODE FOR $A$ :
    $S \leftarrow []$
    loop: On query $x$
        Append $x$ on $S$ and parse $S$ as $(x_1, \ldots, x_\tau)$
        $state \leftarrow \perp$
        for $i \in [\tau]$ do
            $(y, state) \leftarrow C_i(x_i, state)$
        return $y$
```

(b) Equivalent formulation

Figure 1: Stateful Algorithm with respect to $\{C_i\}_{i \in \mathbb{Z}}$

DMAC schemes can be trivially executed in parallel. Note that as straightforward as this might be in the classical setting, quantum interference requires special treatment. Nonetheless, as our DMAC construction is effectively extracted from the one-time quantum signature from [12], the proof follows immediately from their reduction of multi-bit to single-bit one-time quantum signatures [12, Section 5].

Theorem 1. *There exists a secure DMAC in the plain model.*

In terms of quantum resources, the construction of Aaronson and Christiano [3] achieves a single-bit DMAC using $O(n)$ qubits, where n is the security parameter. Thus, to achieve an $l-$bit DMAC we need $O(nl)$ qubits.

4 Reducing stateful to stateless oracles

Here we show that the notion of quantum DMACs is powerful enough to turn any stateful and classically-queried classical oracle into a stateless one. Our technique follows the approach of Döttling et al. [13]. We model a stateful oracle as a stateless oracle together with a stateful database that stores the queries. Then every time the oracle is queried, it is reset and then runs all the previous queries, followed by the last one.

Using this formalization the transformation of a stateful algorithm A into a stateless B works as follows. As a first step assume some polynomial number q of queries are allowed. We create q single-bit DMAC key-pairs (sk_i, ρ_i). Then the algorithm B has the following structure. At the first time it is called, it is queried with x_1 together with a tag t_1 on x_1 with respect to the key sk_1. If the tag is valid, then the algorithm runs as a subroutine A with input x_1 and returns A's output. For the next query x_2, the calling algorithm should provide both (x_1, t_1) and (x_2, t_2), where t_2 is a tag of x_2 with respect to sk_2. Now B will first run A on the first input and then run A on the second input and return this result.

Stateful oracle. A stateful oracle can be thought of as a sequence of stateless oracles $\{C_i\}$, where each of them after execution outputs a state that is fed as input to the next oracle together with a query. Equivalently, a stateful oracle could keep a list of all the previous queries and re-execute the whole computation from the beginning for each new query.

Definition 3 (Stateful algorithm)**.** *A stateful oracle A with respect to a family of stateless oracles* $\{C_i\}_{i \in \mathbb{Z}}$ *works as shown in Figure 1a.*

Up to a polynomial slowdown, an equivalent formulation of a stateful oracle is shown in Figure 1b.

Stateful to stateless transformation A stateful to stateless oracle transformation is an algorithm that takes as input the description of a stateful oracle and returns the description of a stateless oracle together with a quantum state. We require the correctness that any algorithm with oracle access to the stateful algorithm can be simulated by another algorithm with oracle access to the stateless one. We also require

```
$B_{(s_1,\ldots,s_q,i)}((x_1,t_1),\ldots,(x_\tau,t_\tau))$
    if $\mathsf{DMAC.Ver}(s_j,x_j,t_j)=0$ for some $j\in[\tau]$ then
        return "Invalid tag"
    $state \leftarrow \perp$
    for $j\in[\tau]$ do
        $(y, state) \leftarrow C_{i,j}(x, state)$
    return $y$
```

Figure 2: The class of stateless oracles $\mathcal{B}$

the security that an algorithm with access to the stateless oracle does not have extra power over one that has access to the stateful one.

Definition 4 (Stateful to Stateless transformation). *Let $\mathcal{A}$ be a family of stateful oracles $\{A_i\}_{i\in\mathbb{Z}}$.* Gen *is a stateful to stateless oracle transformation with respect to $\mathcal{A}$ if there is a family $\mathcal{B}=\{B_j\}_{j\in\mathbb{Z}}$ of stateless oracles such that:*

- $\mathsf{Gen}(1^n,i)\rightarrow(\rho,j)$ *is an algorithm that takes as input a security parameter n as well as an index i that corresponds to the stateful oracle A_i and returns a quantum state ρ together with an index j that corresponds to the stateless oracle B_j.*

The transformation has to satisfy the following properties:

Completeness. *For any (polynomial time) algorithm C, there exists a (respectively polynomial time) simulator S such that for any $i\in\mathbb{Z}$, $C^{A_i}\equiv S^{B_j}(\rho)$, where $(j,\rho)\leftarrow\mathsf{Gen}(1^n,i)$.*

Security. *For any (polynomial time) algorithm C, there exists a (respectively polynomial time) time simulator S such that for any possibly auxiliary quantum state* aux*, which is a partial system that may be entangled with a reference system R, say $tr_R\widetilde{\mathsf{aux}}=\mathsf{aux}$, and for any $i\in\mathbb{Z}$,*

$$C^{B_j}\otimes\mathbb{I}_R(1^n,\rho\otimes\widetilde{\mathsf{aux}})\sim_s S^{A_i}\otimes\mathbb{I}_R(1^n,\widetilde{\mathsf{aux}}),$$

where $(j,\rho)\leftarrow\mathsf{Gen}(1^n,i)$.

Here tr_R denotes the partial trace with respect to system R and $\mathbb{I}_R$ denotes the identity on system R.

4.1 The transformation

Here we formally present the construction that transforms any polynomial time stateful oracle into a stateless one. Intuitively, the construction works as follows. Our new stateless oracle B has to take as input all the previous queries. In this way, we guarantee that B does not need to keep a state. On the other hand, we have to impose that B cannot be rewound, i.e., if the first query is x, then there is no way we can start B from the beginning with a query $x'\neq x$. To achieve this, B is parameterized by a list $s_1,\ldots,s_q$ of secret keys for a DMAC, where q is the total number of queries. For each query x_j, the calling algorithm has to also provide a tag t_j for x_j corresponding to the secret key s_j. Before executing the query, B first verifies that the tags for all the queries are valid. If this is the case, then it runs all the queries one by one and returns the final outcome.

Let $\mathcal{A}=\{A_i\}_{i\in\mathbb{Z}}=\{C_{i,j}\}_{i,j\in\mathbb{Z}}$ be the class of all polynomial time oracles, where $C_{i,j}$ are the stateless oracles corresponding to A_i. Moreover, let $(\mathsf{DMAC.Gen},\mathsf{DMAC.Tag},\mathsf{DMAC.Ver})$ be a secure DMAC. We define the class $\mathcal{B}=\{B_{(s_1,\ldots,s_q,i)}\}_{s_1,\ldots,s_q,i}$ in Figure 2.

Clearly, $\mathcal{B}$ is a class of stateless oracles. Now, the generation algorithm $\mathsf{Gen}(1^n,i)$ first runs $(s_j,\rho_j)\leftarrow\mathsf{DMAC.Gen}(1^n)$ for each $j\in[q]$ and then returns $((s_1,\ldots,s_q,i),\rho_1\otimes\ldots\otimes\rho_q)$.

To argue completeness, let C be any algorithm that has access to the stateful oracle A. We will create a simulator S that takes as input the quantum state $\rho_1\otimes\ldots\otimes\rho_q$ and has oracle access to the stateless oracle

B. S initializes $\tau = 0$ and the sequence $\mathcal{S}$ to be the empty sequence. Then it starts C and simulates C's oracle as shown in Figure 3.

```
ORACLE($x$)
  $\tau \leftarrow \tau + 1$
  $t \leftarrow \mathsf{DMAC.Tag}(\rho_\tau, x)$
  Append $(x, t)$ on $\mathcal{S}$ and parse $\mathcal{S}$ as $((x_1, t_1), \ldots, (x_\tau, t_\tau))$
  return $B((x_1, t_1), \ldots, (x_\tau, t_\tau))$
```

Figure 3: The oracle that S simulates

Therefore, the completeness follows from that of the DMAC. We include the proof of security in Appendix B.

In terms of quantum resources, we need a DMAC for each state of the stateful oracle. Thus, for an oracle that accepts queries of length l and has m states, we need $O(lmn)$ qubits, where n is the security parameter. In particular, for one-time programs we need $O(ln)$ qubits and for one-time memories that store 2 secrets, we need $O(n)$ qubits (the query length is a single bit determining which of the two secrets we want to retrieve).

5 Oracle one-time memories

In this section we provide a definition of OTMs relative to an oracle. It is well known that it is impossible to achieve OTM even in the quantum world and even with computational assumptions in the plain model (without an oracle). In Appendix C we define one-time programs (OTP) again relative to an oracle.

An OTM is a memory that stores k secrets $s_1, s_2, \ldots, s_k$ but only one can be extracted. OTMs were first defined by Goldwasser and Rothblum [22] in order to construct OTPs. OTMs have then been studied extensively in the quantum setting, with Broadbent et al. [10] presenting a construction of quantum OTPs from quantum OTMs. Liu [27] has shown that OTMs are possible in the isolated qubits model, where each single qubit is manipulated by an adversary and those adversaries are allowed to communicate only classically.

(Classical) oracle OTMs. Our definition below is inspired by the definition of Broadbent et al. [8].

Definition 5 (Classical-oracle one-time memories). *Let $\mathcal{C} = \{C_j\}_{j \in \{0,1\}^*}$ be a family of polynomial-sized classical circuits. A $\mathcal{C}$-oracle one-time memory ($\mathcal{C}$-OTM) scheme is a pair of polynomial-sized quantum algorithms* $(\mathsf{Gen}, \mathsf{Extract})$ *with the following properties:*

- $\mathsf{Gen}(1^n, s_1, s_2, \ldots, s_k) \to (\rho, j)$ *is an algorithm that takes as input k secret classical bit-strings $s_1, s_2, \ldots, s_k$, each of length n, and outputs a quantum state ρ that intuitively encodes the k secrets. In addition, it outputs an index j corresponding to the circuit C_j.*
- $\mathsf{Extract}^C(\rho, i) \to s$ *is an oracle algorithm that takes as input an index i and a quantum state ρ and makes a single oracle query to a circuit C. It outputs a classical bit-string s.*

A $\mathcal{C}$-OTM satisfies the following security properties.

Completeness. *For any $s_1, s_2, \ldots, s_k \in \{0,1\}^n$, for any index i and for any (ρ, j) that is output by* $\mathsf{Gen}(1^n, s_1, s_2, \ldots, s_k)$, *it holds that* $\mathsf{Extract}^{C_j}(\rho, i) = s_i$.

One-timeness. *For any (possibly unbounded) adversary A there exists a (respectively unbounded) simulator S, such that for any k bit-strings $s_1, s_2, \ldots, s_k$ of length n and for any auxiliary (mixed) quantum state* aux,

$$A^{C_j}(1^n, \rho \otimes \mathsf{aux}) \sim_s S^{\mathsf{OT}_{s_1, s_2, \ldots, s_k}}(1^n, \mathsf{aux}),$$

where $(\rho, j) \leftarrow$ $\mathsf{Gen}(1^n, s_1, s_2, \ldots, s_k)$ *and the distributions are over the coins of A, S, and* Gen. *The oracle* $\mathsf{OT}_{s_1, s_2, \ldots, s_k}$ *on input i returns s_i and then halts.*

Similarly to the original terminology, we will refer to ρ derived as above, *as a $\mathcal{C}$-OTM with contents $s_1, s_2, \ldots, s_k$ and oracle $C_j \in \mathcal{C}$.* We will also use $\mathsf{OTM}(s_1, s_2, \ldots, s_k)$ to denote the corresponding pair (ρ, C_j).

Notice that this primitive is impossible to achieve in the classical world. Indeed, an adversary A with oracle access to C, given a classical bitstring ρ, can do the following trivial attack. First, copy the ρ into a new register ρ'. Then, run $\mathsf{Extract}^C(\rho, 0)$ and $\mathsf{Extract}^C(\rho', 1)$ and return the two results. Clearly, this cannot be simulated by S.

On the other hand, currently it is not clear whether this primitive is achievable in the quantum world. Now A cannot simply undo the computation since the oracle C works only classically and A has only oracle access to it.

Remark: The above definition seems to be the most natural way to model classical stateless tokens. In practice the above definition can be instantiated by having a programmable trusted hardware that runs some code based on some secret information. For example, smartcards or Intel's SGX could be a possible way to achieve such a hardware. Notice that obfuscating such an oracle and giving it to the adversary is not secure: since the adversary is given actual (albeit obfuscated) code in the form of a description of a quantum circuit with elementary quantum gates, it is always possible to reverse the gates and thus the whole computation.

6 One-time Backdoored Devices

In this section we demonstrate how one-time backdoors can provide a middle-ground solution to the smartphone conundrum of privacy vs. law enforcement, thus addressing the original problem that motivated this paper. The original motivation is to create a system that allows a device (e.g., smartphone) vendor to embed in its devices a content locking mechanism, and create (and locally store) a disposable unlocking backdoor. The system should satisfy the following properties:

Setup. There should be a setup algorithm that creates the code for the locking device and the relevant unlock backdoor.

Confidentiality. No one (in particular, no PPT adversary) should be able to extract any information from the locked device without the backdoor. This should be true even if the adversary has (partial) knowledge about the keys and/or states of the unlocked devices and about the state of the locked devices.

One-time unlock. Using the unlocking backdoor, the vendor should be able to unlock exactly one phone. In particular, it should not be able to use the backdoor to extract information from two locked devices. This should, again, be true even if the adversary has (partial) knowledge about the keys and/or states of the unlocked devices and about the state of the locked devices.

(Non-interactive) Extendibility. The vendor should be able to program more (new) devices to be unlockable with the disposable backdoor without resetting the entire system, or, in particular, interacting with the devices that are already set up and distributed.

The above can be achieved by having a quantumly enabled vendor equipped with a stateless token and a classical set of devices. Note that we assume classical devices as it is unlikely that current technology will yield hand-held devices with quantum storage and computation capabilities any time soon. In addition, we do not assume that each device has a secure storage or trusted-hardware module. In fact, the adversary in our definition has full access on the state of the locked device and in particular could copy this state into a quantum computer, and perform a quantum attack. Looking ahead, such an attack will fail because the one-time memory where the relevant backdoor information is stored involves a token that can only be classically queried.

Our system works as follows :

The Setup Algorithm: Let N be the number of initial devices, denoted by $D_1, \ldots, D_N$ that the vendor wishes to set up, and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ denote the key-generation, encryption, and decryption algorithms for a symmetric-key encryption scheme. Without loss of generality, we will assume the scheme to be IND-CPA-secure [4] as this will already provide us with the desirable confidentiality. Of course one can consider schemes with higher level of security, e.g., IND-CCA-security, if an application needs additional security guarantees. The vendor V performs the following steps to set up all the N devices:

1. The vendor V uses the key generation algorithm Gen N times to generate N independent n-bit secret keys $k_1, \ldots, k_N$ (where n is the security parameter).

2. The code of each D_i contains the following locking procedure: D_i has the key k_i locally stored; to lock itself—e.g., if its user inputs incorrect pins too many times—D_i uses Enc to encrypt its state with key k_i and erases the key k_i. Without loss of generality, we assume that in the locked state, the phone might accept a command to output its encrypted state.[4]

3. V creates a 1-out-of-N OTM that encodes the keys $k_1, \ldots, k_N$. Subsequently, the vendor erases the keys $k_1, \ldots, k_N$ from its local state (so that they only reside in the OTM) and also the coins used in their generation. Notice that, after the phone locks itself, the key is only available through the OTM, and even the vendor cannot extract the encryption key.

UNLOCKING A DEVICE WITH THE BACKDOOR: The vendor (or anyone in possession of the OTM) can use the OTM to unlock any locked device in a straightforward manner: To unlock D_i, the vendor extracts the key k_i from the OTM and uses it to decrypt the state.

One can easily confirm the security of OTMs and the encryption scheme ensures that our protocol satisfies the properties required above, i.e., setup, confidentiality, and the one-time-unlock property: The fact that the setup algorithm achieves the setup guarantees follows directly by inspection of the protocol. Confidentiality follows directly from the CPA-security of the encryption scheme. Finally, the one-time-unlock property follows from the CPA-security of the encryption scheme and the security of the OTMs. Note that OTMs are assumed secure even with respect to any auxiliary information. Hence, (partial) knowledge about the state/keys of the unlocked devices or the state of the locked devices does not help the adversary to learn any information from the OTM about any key (or about the corresponding encrypted states) other than the extracted.

6.1 Extendable OTMs

To complete our analysis, we need to describe how to obtain non-interactive extendibility. In order to do this, we define in the following an extendable version of OTMs, which are memories that can be extended by adding more secrets into them. Using such memories, one can trivially add new devices in the system without interacting with existing devices by simply running the setup algorithm for the new devices and adding the new keys to the existing OTM, instead of storing them in a new OTM. We achieve this by having n (1-out-of-2) OTM for n secrets. Each (1-out-of-2) OTM i encodes two values. First, a random key k_i. Second, the actual secret s_i xor'ed with all the previous keys $k_1 \oplus \ldots \oplus k_{i-1}$. Informally, in order to extract the secret s_i, we first need to extract all the keys k_j for $j < i$, thus destroying all the previous OTM. In terms of quantum resources, we need $O(n)$ qubits for each (1-out-of-2) OTM and thus $O(i \cdot n)$ qubits for a (1-out-of-i) OTM. We present the construction formally in the following.

Extendable OTMs are OTMs that can be encapsulated with additional secrets. Correctness should guarantee that any of the up-to-now secrets encoded can be extracted, whereas security should guarantee that only one of these secrets can be extracted. In this section we will omit the oracles in notation to simplify the presentation.

Definition 6 (Extendable one-time memories). *An extendable one-time memory (EOTM) is an OTM augmented with an extra algorithm* Extend *as follows:*

- $\mathsf{Extend}(1^n, \rho, s) \to \rho'$ *is an algorithm that takes as input a quantum state ρ encoding some secrets and a classical bit-string s of length n, and returns a quantum state ρ' encoding the previous secrets plus s.*

The completeness and security extend trivially from the definitions of OTMs. In particular, in this case we want that for any algorithm A that takes as input an EOTM encoding k secrets $s_1, \ldots, s_k$ to have a simulator that can compute anything that A can compute but with only oracle access to $\mathsf{OT}_{s_1,\ldots,s_k}$, where $\mathsf{OT}_{s_1,\ldots,s_k}$ takes as input an index i, returns s_i and then halts.

[4]It is assumed that the vendor can extract the encrypted state from the phone's storage, anyway.

6.2 A black-box construction of EOTMs

Our construction turns any OTM into an EOTM in a black-box manner. We give the intuition with an example below that illustrates how to go from an OTM that encodes two secrets (the first line) to one that encodes three secrets (the second line). We denote by $\mathsf{OTM}(s_0, s_1)$ the outcome of an one-out-of-two OTM generation algorithm with input s_0, s_1.

$$\mathsf{OTM}(s_1, k_1), \mathsf{OTM}(s_2 \oplus k_1, k_2), k_1 \oplus k_2$$
$$\mathsf{OTM}(s_1, k_1), \mathsf{OTM}(s_2 \oplus k_1, k_2), \mathsf{OTM}(s_3 \oplus k_1 \oplus k_2, k_3), k_1 \oplus k_2 \oplus k_3$$

One can see that we use the classical value of an OTM as a mask for the next secret and then we update the classical value by XORing it with a new key. Notice that in order to extract the value, say s_3, we need to extract both k_1 and k_2 from the previous OTMs and thus we do not have the option to extract any of the other secrets.

Formally, let $(\mathsf{OTM.Gen}, \mathsf{OTM.Extract})$ be an one-out-of-two OTM. In the following we omit the security parameter input 1^n for simplicity. We create an EOTM $(\mathsf{Gen}, \mathsf{Extend}, \mathsf{Extract})$ as shown in Figure 4.

```
$\mathsf{Gen}(s_1, s_2)$
    $k_1, k_2 \leftarrow \{0,1\}^n$
    **return** $\mathsf{OTM.Gen}(s_1, k_1) \otimes \mathsf{OTM.Gen}(s_2 \oplus k_1, k_2), k_1 \oplus k_2$

$\mathsf{Extend}((\rho, k), s)$
    $k' \leftarrow \{0,1\}^n$
    **return** $\rho \otimes \mathsf{OTM.Gen}(s \oplus k, k'), k \oplus k'$

$\mathsf{Extract}((\rho_1 \otimes \ldots \otimes \rho_i, \_), j)$
    $k \leftarrow 0^n$
    **for** $l = 1$ to $j - 1$ **do**
        $k \leftarrow k \oplus \mathsf{OTM.Extract}(\rho_l, 1)$
    **return** $\mathsf{OTM.Extract}(\rho_j, 0) \oplus k$
```

Figure 4: Extendable OTM construction

Theorem 2. *The construction above is an extendable OTM.*

Proof. Any algorithm with access to the OTM and an auxiliary input aux can be simulated, using a straightforward hybrid argument, by one algorithm that has oracle access to the respective one-time OT oracles and is also given aux as input. Moreover, any such algorithm S can be easily simulated by an algorithm S' that has oracle access to the algorithm $\mathsf{OT}_{s_1,\ldots,s_n}$ (that on input i returns s_i and then halts) as follows. S' on input aux, starts S with input aux. If S queries oracle i with bit $b = 1$, S' returns a random key k_i. If S queries oracle i with bit $b = 0$, then S' queries its oracle with value i. Upon getting answer s_i, it returns the value $s_i \oplus (\bigoplus_{j<i} k_j)$, by fixing at random all the keys k_j, $j < i$ that have not been queried. □

Note that it is mandatory for the previous values of k to be erased and only the final one is kept. Indeed, if an adversary has continuous access to the previous classical values of the OTM, it can retrieve all the classical keys $k_1, \ldots, k_\ell$ without destroying the OTMs and thus it can retrieve all the secrets by always extracting the first part of the OTM and XORing it with the respective secret.

7 Further applications

Our stateful-to-stateless reduction has further applications in several other cryptographic areas.

Encryption with Disposable Decryption backdoor In Appendix D we show how to extend the one-time backdoor paradigm to standard cryptographic primitives. Concretely, we look at the case of public-key encryption schemes with disposable decryption-backdoors and we show how we can construct them. We believe that cryptography with disposable backdoors can also be generalized to most cryptographic primitives with game-based security definitions. We leave considering such an extension and investigate further applications/implications as an open research direction.

Physical computation. Fisch, Freund, and Naor [16, 17] introduced the notion of physical computation as a way for mutually untrustful parties to compute functions of physical inputs without revealing these inputs in the clear. This notion generalizes and formalizes the work by Glaser, Barak, and Goldston [20] which suggested applying zero-knowledge for proving that a nuclear weapon is authentic without revealing sensitive information about its design. As a necessary tool for the construction in [16, 17], they use a primitive called "disposable circuit". These are hardware tokens that can be completely destroyed after one computation. In particular, this is a generalized version of tamper-proof tokens, where the tokens compute some function of their input together with some hardwired secret. Using our transformation, one can directly extend the above results and simply use any tamper-proof token to achieve a disposable circuit in the quantum world.

Copy protection and digital rights managements (DRM). Persistent online authentication is a DRM technique where a software can only work as long as it is continuously connected to an online server. Such servers ask from the software to send some authenticated data and then respond with some data that allow the software to run. To eliminate the need for continuous communication, one could hardwire such a server into a (stateful) embedded device. However, such a system would be susceptible to resetting attacks. Using our transformation, one can, in theory, hardwire such a server into a stateless smartcard distributed to the users, and only periodically communicate the appropriate qubits with the users, thus achieving security without continuous connection.

References

[1] Scott Aaronson. Limitations of quantum advice and one-way communication. In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 320–332. IEEE, 2004.

[2] Scott Aaronson. Quantum copy-protection and quantum money. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 229–242. IEEE, 2009.

[3] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 41–60, New York, NY, USA, 2012. ACM.

[4] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 26–45, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[5] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 409–426, 2006.

[6] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, page 175, 1984.

[7] Charles H Bennett, Gilles Brassard, Seth Breidbart, and Stephen Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology*, pages 267–275. Springer, 1983.

[8] Anne Broadbent, Sevag Gharibian, and Hong-Sheng Zhou. Quantum one-time memories from stateless hardware. Cryptology ePrint Archive, Report 2015/1072, 2015.

[9] Anne Broadbent, Sevag Gharibian, and Hong-Sheng Zhou. Towards quantum one-time memories from stateless hardware. *arXiv preprint arXiv:1810.05226*, 2018.

[10] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs - (extended abstract). In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 344–360, 2013.

[11] James B. Comey. Transcripts from a public speech, Brookings Institution, Washington, D.C. `https://www.fbi.gov/news/speeches/going-dark-are-technology-privacy-and-public-safety-on-a-collision-course`.

[12] Shalev Ben David and Or Sattath. Quantum tokens for digital signatures. *arXiv preprint arXiv:1609.09047*, 2016.

[13] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. From stateful hardware to resettable hardware using symmetric ssumptions. In *International Conference on Provable Security*, pages 23–42. Springer, 2015.

[14] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. General statistically secure computation with bounded-resettable hardware tokens. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 319–344. Springer, 2015.

[15] Electronic Frontier Foundation (EFF). The crypto wars: Governments working to undermine encryption. Technical Report, 2014. `https://www.eff.org/files/2014/01/03/cryptowarsonepagers-1_cac.pdf`.

[16] Ben Fisch, Daniel Freund, and Moni Naor. Physical zero-knowledge proofs of physical properties. In *International Cryptology Conference*, pages 313–336. Springer, 2014.

[17] Ben A Fisch, Daniel Freund, and Moni Naor. Secure physical computation using disposable circuits. In *Theory of Cryptography Conference*, pages 182–198. Springer, 2015.

[18] Dmitry Gavinsky. Quantum money with classical verification. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 42–52. IEEE, 2012.

[19] Marios Georgiou and Iordanis Kerenidis. New constructions for quantum money. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 44. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[20] Alexander Glaser, Boaz Barak, and Robert J. Goldston. A zero-knowledge protocol for nuclear warhead verification. *Nature*, 510:497–502, 2014. See also article by R. Stone (Science, June 2014).

[21] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2009.

[22] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Annual International Cryptology Conference*, pages 39–56. Springer, 2008.

[23] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[24] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2010.

[25] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128. Springer, 2007.

[26] Arash Khamooshi. Breaking down Apple's iPhone fight with the U.S. Government. The New York Times, 2016. `https://www.nytimes.com/interactive/2016/03/03/technology/apple-iphone-fbi-fight-explained.html?_r=0`.

[27] Yi-Kai Liu. Single-shot security for one-time memories in the isolated qubits model. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 19–36, 2014.

[28] Tomohiro Ogawa and Hiroshi Nagaoka. Making good codes for classical-quantum channel coding via quantum hypothesis testing. *IEEE Transactions on Information Theory*, 53(6):2261–2266, 2007.

[29] Fernando Pastawski, Norman Y Yao, Liang Jiang, Mikhail D Lukin, and J Ignacio Cirac. Unforgeable noise-tolerant quantum tokens. *Proceedings of the National Academy of Sciences*, 109(40):16079–16082, 2012.

[30] Michael S. Schmidt and Richard Perez-Pena. F.B.I. treating san bernardino attack as terrorism case. The New York Times, 2015. `https://www.nytimes.com/2015/12/05/us/tashfeen-malik-islamic-state.html`.

[31] Andreas Winter. Coding theorem and strong converse for quantum channels. *IEEE Transactions on Information Theory*, 45(7):2481–2485, 1999.

A DMAC construction

Here we include the construction of disposable message-authentication codes as defined by Ben-David and Sattath [12] which builds upon the construction of Aaronson and Christiano [3]. Let H be the n-qubit Hadamard operator. For a subspace $A \subseteq \mathbb{F}_2^n$, let $A^\perp$ be its orthogonal complement.

```
Gen($1^n$)
    Pick a random subspace $A \subseteq \mathbb{F}_2^n$ of dimension $n/2$ uniformly over all the $(n/2)$-dimensional subspaces of $\mathbb{F}_2^n$
    Let $\rho = |\psi\rangle\langle\psi|$, where $|\psi\rangle = \frac{1}{\sqrt{|A|}} \sum_{\mathbf{v}\in A} |\mathbf{v}\rangle$, that is, $|\psi\rangle$ is a uniform superposition of all the vectors in $A$.
    **return** $(A, \rho)$
Tag($\rho, b$)
    **if** $b = 0$ **then**
        Measure $\rho$ in the computational basis and **return** the outcome
    **else**
        Measure $H\rho H^\dagger$ in the computational basis and **return** the outcome
Ver($A, b, \mathbf{v}$)
    **if** $b = 0$ **then**
        **return** $\mathbf{v} \in^? A$
    **else**
        **return** $\mathbf{v} \in^? A^\perp$
```

The completeness of the scheme follows easily from [3, Lemma 21]. The security of the scheme follows directly from [12, Theorem 16]. In particular, it is proved that any (even computationally unbounded) quantum adversary that is given as input ρ, needs an exponential number of queries to the verification oracle in order to find a vector in A and a vector in $A^\perp$.

B Security analysis of stateful-to-stateless transformation

To argue security, we create a simulator S that takes as input an auxiliary state aux and has oracle access to the algorithm A. S first creates q pairs of DMAC keys $s_1, \ldots, s_q$ together with their quantum states $\rho_1, \ldots, \rho_q$. Then S starts $C \otimes \mathbb{I}_R$ with input $\rho_1 \otimes \ldots \otimes \rho_q \otimes \widetilde{\mathsf{aux}}$, where $\mathrm{tr}_R \widetilde{\mathsf{aux}} = \mathsf{aux}$. Moreover, S simulates the oracle B as shown in Figure 5. During the simulation, S initializes two empty lists $\mathcal{Q}, \mathcal{A}$ whose size increases at the same time. Informally, $\mathcal{Q}$ will contain the longest sequence of queries $x_1, \ldots, x_{|\mathcal{Q}|}$ that have a valid tag. $\mathcal{A}$ will contain the corresponding answers that the algorithm A replies. We denote by $\mathcal{Q}_i$ the i-th element of $\mathcal{Q}$ and similarly for $\mathcal{A}$.

```
$B_{\mathsf{sim}}((x_1, t_1), \ldots, (x_\tau, t_\tau))$
    **if** $\mathsf{DMAC.Ver}(s_j, x_j, t_j) = 0$ for some $j \in [\tau]$ **then**
        **return** “Invalid tag”
    **if** $(x_1, \ldots, x_\tau)$ is a prefix of $\mathcal{Q}$ **then**
        **return** $\mathcal{A}_\tau$ (no need to query $A$)
    **if** $\mathcal{Q}$ is not a prefix of $(x_1, \ldots, x_\tau)$ **then**
        **return** $\perp$
    $l \leftarrow |\mathcal{Q}| + 1$
    **for** $i \in [l, \tau]$ **do**
        $\mathcal{Q}_i \leftarrow x_i$ (i.e. append $x_i$ to $\mathcal{Q}$)
        $\mathcal{A}_i \leftarrow A(x_i)$ (i.e. append the answer to $\mathcal{A}$)
    **return** $\mathcal{A}_\tau$
```

Figure 5: The oracle B_{sim} that S simulates

Note that if the execution reaches the line **return** $\perp$, then the adversary will be able to tag two messages using the same key.

Notice that if C is computationally unbounded but limited to a polynomial number of queries, then S has to be also computationally unbounded.

Let E be the event that the line **return** $\perp$ is executed. Let q' be the number of queries C makes to its oracle B and let also $\{(x_{1j}, t_{1j}), \ldots, (x_{\tau_j j}, t_{\tau_j j})\}_{j \in [q']}$ be the queries. Equivalently, this event can be defined as the event that C makes two queries with different messages in some position i and the corresponding tags are both valid:

$$\begin{aligned} E = \{ & \exists j, j' \in [q'], i \in [q] : x_{ij} \neq x_{ij'} \\ & \wedge \mathsf{DMAC.Ver}(s_i, x_{ij}, t_{ij}) = 1 \\ & \wedge \mathsf{DMAC.Ver}(s_i, x_{ij'}, t_{ij'}) = 1 \}. \end{aligned}$$

Then, our simulator works exactly as C except for the event E; i.e., for any output o, any $n \in \mathbb{Z}$ and any auxiliary quantum state aux, it holds that

$$\left| \Pr[C^{B_{(s_1, \ldots, s_q, i)}} \otimes \mathbb{I}_R(1^n, \rho_1 \otimes \ldots \otimes \rho_q \otimes \widetilde{\mathsf{aux}}) = o] - \Pr[S^{A_i} \otimes \mathbb{I}_R(1^n, \widetilde{\mathsf{aux}}) = o] \right| \leq \Pr[E].$$

Now, suppose that there exists an adversary C, value $n \in \mathbb{Z}$ and quantum state aux such that $\Pr[E] \geq e(n)$ for some non-negligible function e. We use C to create an adversary C' against the DMAC. C' takes as input a quantum state ρ and has oracle access to the algorithm $V(\cdot, \cdot)$. It starts by picking a random position $i^* \leftarrow [q]$. In this position, C' will plug in the quantum state ρ. For simplicity we rename ρ as ρ_{i^*}. Moreover, C' creates $q - 1$ pairs $(s_i, \rho_i) \leftarrow \mathsf{DMAC.Gen}(1^n)$ for $i \in [q] - \{i^*\}$. Then $C' \otimes \mathbb{I}_R$ runs $C \otimes \mathbb{I}_R$ with input $(\rho_1 \otimes \ldots \otimes \rho_q \otimes \widetilde{\mathsf{aux}})$ and simulates the oracle B as shown in Figure 6. As before C' has to keep two lists $\mathcal{Q}, \mathcal{A}$ that are initialized to the empty lists.

Informally, C' runs by simulating the stateless oracle and at the same time looking for a pair of inputs that can break the challenge DMAC. For the queries that do not correspond to i^*, C can use its own secret

```
$B_{\mathsf{sim}}((x_1, t_1), \ldots, (x_\tau, t_\tau))$
    **if** $\mathsf{DMAC.Ver}(s_j, x_j, t_j) = 0$ for some $j \in [\tau] - \{i^*\}$ **then**
        **return** "Invalid tag"
    **if** $i^* \leq \tau$ and $V(x_{i^*}, t_{i^*}) = 0$ **then**
        **return** "Invalid tag"
    **if** $(x_1, \ldots, x_\tau)$ is a prefix of $\mathcal{Q}$ **then**
        **return** $\mathcal{A}_\tau$ (no need to query $A$)
    **if** $\mathcal{Q}$ is not a prefix of $(x_1, \ldots, x_\tau)$ and $x_{i^*} \neq \mathcal{Q}_{i^*}$ **then**
        Stop simulation and **return** $(x_{i^*}, t_{i^*}, \mathcal{Q}_{i^*}, t^*)$
    **else**
        Abort
    **if** $i^* \leq \tau$ **then**
        $t^* \leftarrow t_{i^*}$ (remember the first tag)
    $l \leftarrow |\mathcal{Q}| + 1$
    **for** $i \in [l, \tau]$ **do**
        $\mathcal{Q}_i \leftarrow x_i$ (i.e. append $x_i$ to $\mathcal{Q}$)
        $\mathcal{A}_i \leftarrow A(x_i)$ (i.e. append the answer to $\mathcal{A}$)
    **return** $\mathcal{A}_\tau$
```

Figure 6: The oracle B_{sim} simulated by C'.

key. For the ones that correspond to i^*, the simulator uses its verification oracle V. If the adversary ever submits two different sequences of queries such that they are not a prefix of each other, then the simulation stops. With probability $1/q$ the sequences will differ on the i^*-th position, in which case C' will be able to break its challenge.

We can see that $\mathsf{Adv}_{C'}^{\mathsf{DMAC}} = \Pr[E]/q \geq e(n)/q$, which implies that C' breaks the DMAC game with non-negligible probability by using only polynomially many queries to the verification oracle.

C One-Time Programs

In this section we define one-time programs (OTP) [22, 10] relative to an oracle and we show formally how we can build them using our stateful-to-stateless oracle transformation. As they are a generalization of OTMs, OTPs are also impossible in the quantum plain model (without oracles).

(Classical) oracle one-time programs. In the following we define oracle OTPs, which are programs that can be run exactly once. To overcome the impossibility, we allow OTPs access to a classical oracle with classical interface. Our definition is inspired by the definition of Broadbent et al. [8].

Definition 7 (Classical Oracle one-time Programs)**.** *Let $\mathcal{C} = \{C_j\}_{j\in\{0,1\}^*}, \mathcal{C}' = \{C'_{j'}\}_{j'\in\{0,1\}^*}$ be two classes of polynomial-sized classical circuits. A $(\mathcal{C}, \mathcal{C}')-$one-time program (denoted as $(\mathcal{C}, \mathcal{C}')-$OTP) is a pair of algorithms* $(\mathsf{Gen}, \mathsf{Extract})$ *with the following properties:*

- $\mathsf{Gen}(1^n, j) \rightarrow (\rho, j')$ *is an algorithm that takes as input a security parameter n and an index j and outputs a quantum state ρ and an index j'.*
- $\mathsf{Extract}^C(\rho, x) \rightarrow y$ *is a quantum algorithm that takes as input a quantum state ρ and a classical input x and has oracle access to a circuit C. It outputs a bit-string y.*

An OTP satisfies the following two properties.

Completeness. *For any $n \in Z$, any $j \in \{0,1\}^n$ and any input x, it holds that*

$$\mathsf{Extract}^{C'_{j'}}(\rho, x) = C_j(x),$$

where $(\rho, j') \leftarrow \mathsf{Gen}(1^n, j)$.

Security. *For any (possibly unbounded) adversary A, there exists a (respectively unbounded) simulator S, such that for any $n \in \mathbb{Z}$, any $j \in \{0,1\}^n$ and any quantum auxiliary (mixed) quantum state* aux,

$$A^{C'_{j'}}(1^n, \rho \otimes \mathsf{aux}) \sim_s S^{\mathsf{OT}_j}(1^n, \mathsf{aux}),$$

where $(\rho, j') \leftarrow \mathsf{Gen}(1^n, j)$ *and the distributions are over the coins of A, S, and* Gen. *The oracle* OT_j *on input x returns $C_j(x)$ and then halts.*

C.1 One-time program construction

Using our general transformation of stateful to stateless oracles, it is easy to create OTPs relative to a classical oracle. To see this, notice that any algorithm with oracle access to an OTP can be turned into an algorithm that has oracle access to a stateless version of an OTP together with a quantum state.

In the following we formally prove this idea. The reader can feel free to skip the formal proof without missing important details.

Let $\mathcal{C} = \{C_i\}$ be the class of polynomial-sized classical circuits. Moreover, let OT_i be the one-time version of the circuit C_i, i.e., OT_i on input x returns $C_i(x)$ and then halts. Let Gen' be the transformation from Section 4 with respect to the class $\{\mathsf{OT}_i\}_i$. In other words, Gen' turns the class of stateful algorithms $\{\mathsf{OT}_i\}_i$ into the class of stateless algorithms $\mathcal{B} = \{B_j\}_j$. Let ρ be the quantum state output by Gen'. Let $D_x^{\mathsf{OT}_i}$ be an algorithm with oracle access to OT_i that returns $\mathsf{OT}_i(x)$. In other words, D_x queries its oracle OT_i with the value x and returns the result. By the completeness of the stateful-to-stateless transformation, there exists an algorithm S_x such that

$$D_x^{\mathsf{OT}_i} \equiv S_x^{B_j}(\rho),$$

where $(j, \rho) \leftarrow \mathsf{Gen}'(1^n, i)$. Our goal is to create a $(\mathcal{C}, \mathcal{B})-$OTP $(\mathsf{Gen}, \mathsf{Extract})$. We define the two algorithms in fig. 7.

```
$\mathsf{Gen}(1^n, i)$
    **return** $\mathsf{Gen}'(1^n, i)$
$\mathsf{Extract}^B(\rho, x)$
    **return** $S_x^B(\rho)$
```

Figure 7: One-time Program construction

Notice that the simulator S_x with oracle access to B_j indeed returns the value $C_i(x)$, and hence the completeness follows. One-timeness follows directly from the security of the stateful-to-stateless transformation. For any adversary A there exists a simulator S' such that for any auxiliary quantum state aux, it holds that

$$A^{B_i}(1^n, \rho \otimes \mathsf{aux}) \sim_s S'^{\mathsf{OT}_i}(1^n, \mathsf{aux}).$$

Remark 1 (From single use to limited-use (e.g., constant-times) backdoors)**.** *Our definitions of one-time (and disposable-backdoor) primitives trivially extend to the many-time cases. However, the corresponding constructions are not trivial. In other words, we cannot just create an n-time primitive by outputting n one-time primitive. To see this, consider the case of multi-bit DMACs. If we have two copies of the quantum state for tagging each bit, we can tag both 0 and 1 for each position of a bit-string, and this makes it trivial to tag any bit-string. To overcome this we use again our stateful to stateless oracle reduction. In other words, we can consider a stateful program that allows only n runs and then turn it into a stateless one. Since n-time memories and n-time message authentication codes are a special case of an n-time program, we get the corresponding primitives. In order to keep our constructions easy to read, we restrict this paper to the one-time versions.*

D Encryption with Disposable Decryption backdoor

Our goal in this section is to turn any CCA encryption scheme into a one-time backdoored one in the sense that one can have access to a quantum backdoor that can be used only once to perform decryptions.

First, we give a general encryption definition. We then extend this definition to its one-time backdoored version. The backdoored version should satisfy three properties. First, it should satisfy the correctness of the original scheme. Second, it should satisfy the correctness of the backdoor, i.e., the backdoor correctly decrypts a ciphertext. Last, in the security game, the adversary is given additionally a quantum backdoor and its goal is to now break *two* challenges.

Definition 8 (Encryption)**.** *An encryption scheme consists of three algorithms* $E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

- $\mathsf{Gen}(1^n) \to (\mathsf{ek}, \mathsf{dk})$ *is a key-generation algorithm that takes as input a security parameter* n *and returns an encryption key and a decryption key. In the case of symmetric encryption* $\mathsf{dk} = \mathsf{ek}$*.*
- $\mathsf{Enc}(\mathsf{ek}, m) \to c$ *is the encryption algorithm that takes as input the encryption key and a message* m *and returns a ciphertext* c*.*
- $\mathsf{Dec}(\mathsf{dk}, c) \to m$ *is the decryption algorithm that takes as input the decryption key and a ciphertext and returns a message* m *or* $\perp$*.*

Completeness. *E is complete if*

$$\Pr[\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m)) = m] = 1,$$

where the randomness is over $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^n)$*.*

Security is defined via a code-based game G [5] between a challenger and an adversary A, where A is given access to some oracles as shown below. The adversary begins by calling the oracle INIT, which returns an encryption key. Then the adversary is allowed to call the oracles ENC, DEC and CHAL and in the end it calls the oracle FIN, which finally outputs a bit. For an adversary A, let $\mathsf{Adv}_A^{\mathsf{CCA}} = \left|\Pr[\textsc{Fin} = 1] - \frac{1}{2}\right|$ for the security game defined in Figure 8.

Security. *E is secure if for any polynomial time quantum adversary A it holds that*

$$\mathsf{Adv}_A^{\mathsf{CCA}} \leq \mathsf{negl}(n).$$

```
Oracle Init($1^n$)
    $b \leftarrow \{0,1\}$
    $S \leftarrow \emptyset$
    $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^n)$
    return $\mathsf{ek}$
Oracle Enc($m$)
    return $\mathsf{Enc}(\mathsf{ek}, m)$
Oracle Chal($m_0, m_1$)
    $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
    return $c^*$
Oracle Dec($c$)
    $S \leftarrow S \cup \{c\}$
    return $\mathsf{Dec}(\mathsf{dk}, c)$
Oracle Fin($b'$)
    if $c^* \notin S$ then
        return $b = b'$
    else
        return $\perp$
```

Figure 8: Standard CCA security

The above definition captures both public-key and secret-key encryption. In the case of secret-key encryption, the oracle Enc is not redundant. A more general definition that captures both public-key encryption and identity-based encryption is also possible. Moreover, it could be possible to abstract the definition even more in order to capture signatures or ideally any cryptographic primitives. We reserve such a definition and its corresponding one-time backdoor construction for a future work.

Encryption with disposable backdoors. An encryption scheme with a disposable backdoor is an encryption scheme augmented so that the generation algorithm produces also a quantum backdoor β as well as a description of a classical oracle. Moreover, the scheme has an additional algorithm Rec that uses the backdoor to decrypt a ciphertext.

Definition 9 (Encryption with disposable backdoors)**.** *Let $\mathcal{C} = \{C_k\}_{k\in\{0,1\}^*}$ be a family of polynomial-sized classical circuits. A $\mathcal{C}$-backdoored encryption scheme ($\mathcal{C}$-*Back*) consists of algorithms* $(\mathsf{Gen}, \mathsf{Rec}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

- $\mathsf{Gen}(1^n) \to (\mathsf{ek}, \mathsf{dk}, \beta, k)$ *is a key-generation algorithm that takes as input a security parameter n and returns an encryption-key* ek*, a decryption-key* dk*, a quantum state β, and an index k to a circuit C_k.*
- $\mathsf{Rec}^C(\beta, c) \to m$ *is a "one-time decryption" algorithm that takes as input a backdoor β and a ciphertext c and returns a message m or $\perp$.* Rec *has also oracle access to a classical circuit C.*
- $\mathsf{Enc}(\mathsf{ek}, m) \to c$ *is the encryption algorithm that takes as input the encryption key and a message m and returns a ciphertext c.*
- $\mathsf{Dec}(\mathsf{dk}, c) \to m$ *is the decryption algorithm that takes as input the decryption key and a ciphertext and returns a message m or $\perp$.*

Completeness. $\mathcal{C}$-Back *is complete if*

$$\Pr_{(\mathsf{ek},\mathsf{dk},\beta,k)\leftarrow\mathsf{Gen}(1^n)}[\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m)) = m] = 1,$$

and moreover,

$$\Pr_{(\mathsf{ek},\mathsf{dk},\beta,k)\leftarrow\mathsf{Gen}(1^n)}[\mathsf{Rec}^{C_k}(\beta, \mathsf{Enc}(\mathsf{ek}, m)) = m] = 1.$$

```
ORACLE INIT($1^n$)
    $b_0, b_1 \leftarrow \{0,1\}$
    $S \leftarrow \emptyset$
    $(\mathsf{ek}, \mathsf{dk}, \beta, k) \leftarrow \mathsf{Gen}(1^n)$
    return $(\mathsf{ek}, \beta)$
ORACLE ENC($m$)
    return $\mathsf{Enc}(\mathsf{ek}, m)$
ORACLE CHAL0($m_0, m_1$)
    $c_0^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_{b_0})$
    return $c_0^*$
ORACLE CHAL1($m_0, m_1$)
    $c_1^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_{b_1})$
    return $c_1^*$
ORACLE DEC($c$)
    $S \leftarrow S \cup \{c\}$
    return $\mathsf{Dec}(\mathsf{dk}, c)$
ORACLE C($x$)
    return $C_k(x)$
ORACLE FIN($b_0', b_1'$)
    if $c_0^* \notin S$ and $c_1^* \notin S$ then
        return $b_0 = b_0'$ and $b_1 = b_1'$
    else
        return $\perp$
```

Figure 9: One-time backdoored CCA security

The security game is similar to the original one, with the additional property that the adversary is given one backdoor and has to break two challenges as shown in the game below. Notice that an adversary, who just uses the backdoor to decrypt one challenge and then guesses the other, has a probability of 1/2 to win. Moreover, notice the adversary is also given access to the classical oracle C_k. For an adversary A, let $\mathsf{Adv}_A^{\mathsf{BCCA}} := \left|\Pr[\textsc{Fin} = 1] - \frac{1}{2}\right|$ in the security game defined in Figure 9.

Security. $\mathcal{C}$-Back *is secure if for any polynomial time quantum adversary A it holds that*

$$\mathsf{Adv}_A^{\mathsf{BCCA}} \leq \mathsf{negl}(n).$$

Notice that BCCA security also implies CCA security if we do not give any backdoor to the adversary. Indeed, if an adversary was able to break the CCA security of the scheme, it could be used to devise another adversary that breaks BCCA. This new adversary would first use its backdoor to find the bit b_0 and then use the CCA adversary to find b_1.

D.1 Constructing Encryption with Disposable Backdoor

Here we show how to construct Encryption with disposable Backdoor using oracle OTPs. As we have shown above, oracle OTPs are possible in the plain model. The idea is to use the OTP generation algorithm with input the description of the decryption algorithm. In other words, all we have to do is to create a one-time version of the decryption algorithm and this will be our backdoor. Let $G' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be an IND-CCA secure encryption scheme. Let $\mathcal{D} = \{D_{\mathsf{dk}}\}_{\mathsf{dk}}$ be the class of polynomial-sized circuits such that $D_{\mathsf{dk}}(m) = \mathsf{Dec}'(\mathsf{dk}, m)$. We have shown above how to construct OTPs for any class of polynomial-sized circuits, and thus, in particular for $\mathcal{D}$. Therefore, let $(\mathsf{OTP.Gen}, \mathsf{OTP.Extract})$ be a $(\mathcal{D}, \mathcal{C})-$OTP for some

```
$\mathsf{Gen}(1^n)$
    $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}'(1^n)$
    $(\beta, k) \leftarrow \mathsf{OTP.Gen}(\mathsf{dk})$
    return $(\mathsf{ek}, \mathsf{dk}, \beta, k)$

$\mathsf{Rec}^C(\beta, m)$
    return $\mathsf{OTP.Extract}^C(\beta, m)$

$\mathsf{Enc}(\mathsf{ek}, m)$
    return $\mathsf{Enc}'(\mathsf{ek}, m)$

$\mathsf{Dec}(\mathsf{dk}, c)$
    return $\mathsf{Dec}'(\mathsf{dk}, c)$
```

Figure 10: Encryption with disposable backdoor construction

class of circuits $\mathcal{C} = \{C_k\}_k$. We create a $\mathcal{C}$-backdoored encryption scheme $G = (\mathsf{Gen}, \mathsf{Rec}, \mathsf{Enc}, \mathsf{Dec})$ as shown in Figure 10.

Theorem 3. *G is a $\mathcal{C}-$backdoored encryption scheme.*

Proof. The two completeness properties are trivially satisfied by invoking the completeness property of the original encryption scheme and the completeness property of the OTP.

To argue security, note that an adversary A with a backdoor β can be simulated by a simulator S who has access to an additional stateful oracle that decrypts only once and does not add this ciphertext to the set of queried ciphertexts. In this step, the decryption key dk is considered as the auxiliary state. Call $G1$ the game played by S. Now an adversary S who can win this game can easily be turned into an adversary A' that breaks CCA. A' simulates S's oracles INIT,ENC,DEC by calling its own oracles. A' will pick a random bit b and when S calls its oracle CHAL(b) with messages m_0, m_1, A' will use its own challenge oracle with input m_0, m_1. It will get a ciphertext c^*_b and will forward this answer to S. When S calls its oracle CHAL$(1-b)$ then A' will encrypt at random one of the two messages using its own encryption oracle; call this ciphertext c^*_{1-b}. When S calls its one-time decryption oracle, there are three cases. If S queries c^*_b, then A' will reply either m_0 or m_1 with probability 1/2. If S queries c^*_{1-b}, then A' will reply with the corresponding plaintext since A' knows which message it corresponds to. If S queries any other ciphertext, then A' will use its own decryption oracle. Since A' picks the bit b at random, there is at most 1/2 probability that S will not query its one-time decryption oracle with the ciphertext c^*_b. Finally, when S makes a guess between m_0, m_1, A' will return the same guess.

Suppose that there exists a non-negligible function $e(n)$ such that

$$\mathsf{Adv}_A^{\mathsf{BCCA}} \geq e(n).$$

Then by the security of OTP, for the advantage Adv_S^{G1} of S to win the modified game $G1$, it holds that

$$\mathsf{Adv}_S^{G1} \geq e(n) - \mathsf{negl}(n).$$

Thus the advantage of A' in the CCA game is

$$\mathsf{Adv}_{A'}^{\mathsf{CCA}} \geq \frac{1}{2} \cdot \mathsf{Adv}_S^{G1} \geq \frac{1}{2} \cdot e(n) - \mathsf{negl}(n),$$

which is non-negligible.