Password-authenticated Cryptography from Consumable Tokens

Ghada Almashaqbeh

University of Connecticut, ghada@uconn.edu

Abstract. Passwords are widely adopted for user authentication in practice, which led to the question of whether we can bootstrap a strongly-secure setting based on them. Historically, this has been extensively studied for key exchange; bootstrap from a low-entropy password to a high entropy key securing the communication. Other instances include digital lockers, signatures, secret sharing, and encryption.

Motivated by a recent work on consumable tokens (Almashaqbeh et al., Eurocrypt 2022), we extend these efforts and investigate the unified notion of *password-authenticated cryptography* in which knowing a password allows executing cryptographic functionalities. Our model is resistant to exhaustive search attacks due to the self-destruction and unclonability properties of consumable tokens. We study two directions; the first is password-authenticated delegation of cryptographic capabilities in which a party can delegate her, e.g., signing or encryption/decryption, rights to another such that exercising the delegation requires knowing a password. The second direction is password-authenticated MPC, in which only participants who share the correct password can execute the MPC protocol. In both cases, an adversary who does not know the password can try a few guesses after which the functionality self-destructs.

We formally define the notions above and build constructions realizing them. Our primary goal in this work is examining the power of consumable tokens in building password-authenticated cryptography in terms of viable constructions and supported adversary models, and thus, outlining open problems and potential future work directions.

Keywords: Consumable tokens · Password authentication · Delegation · MPC

1 Introduction

Passwords are extensively used in practice to authenticate users, e.g., email access, online banking, cryptocurrency wallet access and recovery, and many more. Being easy to memorize is among the most attractive features of using passwords, but this comes at the expense of security; the relatively small password space permits variety of passive/active and online/offline attacks that may reveal the password especially a poorly-selected one.

There is always the question of whether we can bootstrap a strongly-secure setting based on passwords. The most famous example is password-authenticated

key exchange (PAKE) [2, 12, 27, 31, 32] in which two parties who only share a password can establish a high-entropy key to secure their communications. Another instance is the notion of digital lockers [18]: for a message m encrypted using a low-entropy key, i.e., a human-generated password, the only way to learn anything about m from its ciphertext is through an exhaustive search over the password space. This concept has also been applied to signatures [16], secret sharing [3, 10, 17, 30], and encryption [13, 19, 20]. In a way, knowing the password authorizes the user to perform a cryptographic functionality, e.g., establishing a session (or encryption) key, encrypting/decrypting m or signing it, and reconstructing a shared secret.

We aim to take this concept further and investigate more general directions on *password-authenticated cryptography* that even resist brute search attacks without adding any interaction or using a strong trust assumption (such as trusted or tamper-proof hardware devices). That is, knowing a password allows the user to execute an arbitrary cryptographic functionality for a limited number of times (without needing some party to rate limit the number of password guesses and without relying on trusted hardware under the assumption that they resist side-channel attacks or reverse engineering). This goal is motivated by a recent work on unclonable polymer-based cryptography [5], which showed how proteins, that enjoy unclonability and destructive measurement features, can be used to build bounded-query memory devices called consumable tokens. Each data retrieval attempt consumes part of the token so that after a few queries the token is fully consumed, or destructed. Two applications of consumable tokens were shown in [5]: bounded-query digital lockers that resist exhaustive search attacks so only a few password guesses can be tried, and $(1, n)$-time programs, which are a relaxed version of k-time programs [25] that do not require the strong one-time memory gadgets usually realized using trusted hardware.

Our goal is to examine the power of consumable tokens in realizing new generalized notions of password-authenticated cryptography in terms of viable constructions and supported adversary models. We believe that this is interesting on both the theory and applied sides; it may ignite new research avenues based on the open problems we define to explore combining other technologies, such as quantum computing, in addressing these problems.

1.1 Contributions

In this paper, we investigate two notions under password-authenticated cryptography targeting delegation and multiparty computation (MPC), and show consumable token-based constructions realizing these notions. In particular, we make the following contributions.

New models: PAD and PAMPC. We define two new models in the password-authenticated setting: password-authenticated delegation (PAD) and password-authenticated MPC (PAMPC). The PAD model resembles a setting in which a party delegates her cryptographic power to another such that knowing a password permits exercising the delegation. Furthermore, this delegation can be exercised for a limited number of times after which the functionality self-destructs.

The PAMPC model captures MPC protocols in which participation, and hence, protocol execution, requires knowing a password. In a sense, it is a form of MPC among friends; outsiders who do not know the password cannot participate. Both of these models are in the bounded-query setting allowing an outsider, who does not know the correct password, to make only a few password guesses. We formally define ideal functionalities for the PAD and PAMPC models capturing the password authentication and bounded-query aspects.

Constructions. We show what constructions realizing these models can be built using consumable tokens and under which adversary models. For PAD, we build a generic construction for delegating any cryptographic functionality by combining $(1, n)$-time programs and bounded-query digital lockers from [5]. This construction, however, relies on a strong assumption—indistinguishability obfuscation ($i\mathcal{O}$). Thus, we also develop customized constructions for signatures and decryption delegation that do not require $i\mathcal{O}$, but they need additional setup such as a public bulletin board and public key infrastructure (PKI). All these constructions are in the malicious adversary model.

For PAMPC, we build a construction for two-party non-interactive MPC (NIMPC). In particular, we extend bounded-query digital lockers to implement a password-authenticated non-interactive oblivious transfer (OT) scheme, and combine that with garbled circuits. This construction is secure against semi-honest insider adversaries, and malicious outsider adversaries (the latter is achieved with a probability based on the number of password guesses these adversaries can make). We show another construction for interactive MPC among multiple parties in the secret sharing paradigm. At a high level, the shares needed to execute the MPC protocol are stored in digital lockers so without the password, parties cannot access these shares to execute the protocol. This construction is secure against malicious adversaries.

Open questions and future directions. We discuss open research problems related to addressing stronger adversary models than what can be achieved using consumable tokens. We also outline some future work directions on a hybrid paradigm combining consumable tokens and quantum computing.

1.2 Related Work

So far, password-authenticated cryptography has been investigated in the context of PAKE, digital lockers, encryption, signatures, and secret sharing, as mentioned earlier. There is a long line of research on PAKE, e.g., [1, 2, 4, 12, 21, 27, 29, 31–33] (an extensive survey can be found in [28]) that generally targets two settings: balanced (or symmetric where the two parties share a password) and augmented (or asymmetric in which a server stores some transformation of a client's password rather than the password itself). These efforts resulted in a variety of protocols differing in their hardness assumptions, efficiency, composability, and other features such as threshold support (to improve resistance to server compromise) and two-factor authentication.

Digital lockers are basically an obfuscation of a multibit-output point function $I_{p,m}$. That is, I outputs a message m only when evaluated over the target point (aka password) p, while it outputs $\perp$ for any point $p' \neq p$. Thus, this obfuscation represents an encryption of m using the password p. Canetti et al. [18] showed how to build digital lockers in the plain model such that the only feasible attack is exhaustive search over the password space. Almashaqbeh et al. [5] built digital lockers using consumable tokens that resist this attack. While other efforts addressed non-malleability of digital lockers [8, 22].

The concept of password-authenticated signatures has been introduced in [16], and it relies on having an online server to aid in signing. The key is distributed among the user device and the server, and signing requires an interactive protocol between these two in which the user must enter a password every time she wants to sign a message.

In password-authenticated secret sharing (PASS), first introduced in [10], a secret is shared among several servers such that the user can reconstruct the shared secret only if she provides the correct password and receives valid responses from at least a threshold number of these servers. Offline dictionary attacks are prevented due to having multiple servers, and the success of online attacks is bounded by the number of reconstruction protocols an attacker can engage in with the servers. Followup work removed restrictions on the password distribution and supported secure composition [17], addressed adaptive server corruptions [15], achieved round optimality [30] and robustness [3], and supported post-quantum security [34].

Password-authenticated public key encryption [13] allows authenticating a public key using a human-generated password in a way that protects against offline dictionary attacks. Thus, any party who wants to encrypt messages under such a public key must present the correct password that the owner of the public key used for authentication. Otherwise, i.e., if an incorrect password is used, the owner cannot decrypt the messages correctly. However, the owner must commit to a password during key generation; changing the password requires changing the public key as well. Password-authenticated searchable encryption [19] allows outsourcing encrypted database and retrieving records if the correct password is presented. The data owner can change the password without re-encrypting the database records. However, to protect against offline dictionary attacks, two non-colluding servers are needed and this relies on employing a PASS scheme to share the searchable encryption (high entropy) key among the two servers. Das et al. [20], on the other hand, introduces a distributed password-authenticated symmetric encryption, which is a modification of PASS to allow deriving multiple secret keys (to be used for encryption) instead of a single key. It is also in the distributed server setting, requiring multiple servers to prevent online and offline attacks against the password.

All of the above are instances of password-authenticated cryptography: knowing the password enables, in PAKE, establishing a high-entropy key between two parties; in digital lockers and encryption, encrypting/decrypting messages; in signatures, signing messages; and in PASS, reconstructing a shared secret. As

discussed above, existing solutions either resorted to trusted hardware or interactive protocols (by having multiple servers) to defend against online and offline dictionary (or exhaustive search) attacks, and those who did not need that either admitted exhaustive search attacks or made changing the password costly. We extend these efforts and demonstrate how to achieve the password-authenticated feature for more general cryptographic functionalities—delegation and MPC—that also resist exhaustive search attacks over the password space without such restrictions. This work is considered an initial step laying down formal notions of these models while exploring feasible constructions in the consumable token model that supports the bounded-query feature.

2 Preliminaries

In this section, we review consumable tokens and their two cryptographic applications (bounded-query digital lockers and $(1, n)$-time programs) introduced in [5], which we employ in our constructions.

2.1 Consumable Tokens

Consumable tokens are memory devices that permit a limited number of data retrieval attempts such that each query consumes part of the token, so after a few queries the token is fully consumed (or self-destructed). These are built using unclonable polymers, in particular proteins, that can be used to store digital data. That is, the message is encoded into a sequence of amino acids, then this sequence is synthesized to produce the protein material. Proteins possess unique features. First, there is no way to clone a protein that is given in a small amount. Second, retrieving the digital data out of a protein is a destructive process (the used lab machinery that reveals the protein's amino acid sequence will destroy the chemical bonds in the sample, thus destroying the protein), and it either outputs the valid sequence or nothing.

The consumable token construction proceeds as follows. Storage begins with synthesising the secret message m into protein. This protein is then connected with a shorter protein sequence called a header, such that this header can be recognized by the matching antibodies. So in a sense the header is the secret token key tied to the secret message m. The protein-header chain is then mixed with a large quantity of decoy proteins which are similarly structured but encode random keys and messages. The vial containing the resulting sample is the consumable token. Retrieving m from the vial requires applying the antibodies that recognize the token key (i.e., the secret header) to pull down the target protein encoding m. After cleaving the header, the target protein is then sequenced (using the lab machinery) and this sequence is then decoded to obtain m. Each data retrieval attempt consumes part of the vial due to the destructive nature of sequencing. After a few attempts (i.e., n queries for a small integer n), the whole token will be consumed, enforcing a bounded data retrieval query notion.

This protocol is generalized to build partially-retrievable memory; a token stores v secret messages each of which is tied to a unique key. The protocol guarantees that even a party who knows the secret keys of these messages can retrieve only n secret messages, where $n < v$.

Almashaqbeh et al. [5] defines an ideal functionality for consumable tokens, denoted $\mathcal{F}_{CT}$. In a nutshell, this functionality is parameterized by n and v mentioned above, and it captures a weak consumable token functionality based on the restrictions imposed by biology. In particular, it admits a power gap between the honest party and the adversary; although the token is engineered to allow one data retrieval query (what an honest would obtain), adversaries may possess more powerful lab machinery that may allow them to perform up to n queries instead of one. Furthermore, due to the physical features of antibodies that may allow retrieving m using a wrong key that is close enough to the correct one, this token has a non-negligible soundness error γ. Thus, to use $\mathcal{F}_{CT}$ in provably secure cryptographic applications, additional cryptographic and algorithmic techniques are needed to amplify the soundness error to negligible.

2.2 Bounded-query Digital Lockers

These are digital lockers that resist exhaustive search attacks over the password space. The construction proceeds as follows: the password p is mapped to a token key at random, then the message m is stored under this key inside a token. The mapping function is public, and so the recipient can use the password to retrieve m. An attacker will be able to perform up to n data retrieval queries, and so try up to n password guesses but not more, while an honest who knows the password can retrieve m using only one query. An ideal functionality for digital lockers is defined, which captures a bounded-query point function obfuscation notion, hence it is denoted as $\mathcal{F}_{BPO}$.

The consumable token-based construction that realizes $\mathcal{F}_{BPO}$ amplifies the soundness error of the token to negligible, so it bounds the success probability of the attacker to negligibly above $n/|\mathcal{P}|$. This relies on using (full threshold) secret sharing and a chaining technique to avoid increasing the number of adversarial queries beyond n. In a nutshell, the message m is shared as $m_1, \ldots, m_u$ and each share is stored in a separate token (so a total of u token are sent). To force the adversary to operate on these tokens sequentially, thus avoid increasing the number of queries despite sending a larger number of tokens, these tokens are chained together. That is, share m_i cannot be retrieved before retrieving all shares $m_1, \ldots, m_{i-1}$. This is done by storing a fresh random string along with the message share in each token, and having the password mapping function for token i require a random string that is the combination of all random strings stored in previous tokens.

2.3 $(1, n)$-time Programs

This is a weaker version of one-time programs built using the real-world consumable tokens rather than one-time memory gadgets [25] realized using tamper-

proof trusted hardware (which are needed even in the quantum model [14]). Here, a secret program, or one with some secret state, can be executed by an honest recipient over only one input, while it can be executed by an adversary over n distinct inputs. This power gap is inherited from consumable tokens as discussed earlier. An ideal functionality for $(1, n)$-time programs is defined, denoted as $\mathcal{F}_{BE}$ where BE stands for program encapsulation.

The construction that realizes $\mathcal{F}_{BE}$ combines $i\mathcal{O}$ and $\mathcal{F}_{CT}$ as follows. A program Prog representing a function f is obfuscated, and each input x in the domain of f is associated with a unique secret message m. The obfuscated program, when queried over x, requires also providing the correct m corresponding to x before outputting $f(x)$. The unique m values are generated using a pseudorandom generator for which Prog stores its secret seed, thus enabling the check over x and m. The sender creates a token containing all the messages corresponding to the input space of f, and sends it along with the obfuscated program to the evaluating party. Thus, for an input x, the recipient uses the corresponding token key to retrieve the message m (the mapping between x and token keys is public). An attacker who might seize the program and the token, will be only able to retrieve up to n messages from the token (corresponding to n distinct inputs), and execute f over these inputs.

Since a consumable token can store a few messages, the construction above works for f with a small domain. To handle functions with exponential size input space, linear error correcting codes are used. That is, x is encoded into a codeword of size ω and the sender sends ω tokens containing unique messages. To execute the program over x, the evaluator first computes the codeword of x and uses the i^{th} symbol in this codeword to determine the token key to be used with the i^{th} token. The obfuscated program now checks that the supplemented messages $m_1, \ldots, m_\omega$ correspond to the valid codeword of x and then outputs $f(x)$. The distance of the linear code is configured in a way to guarantee that despite the additional tokens sent, an adversary can only obtain up to n valid codewords. Furthermore, recall that $\mathcal{F}_{CT}$ has a non-negligible soundness error γ. This error must be first amplified to negligible, which is done using secret sharing; share m into u shares and send each share in a separate token under an independently-selected random token key. Retrieving any m requires retrieving all of its shares, which makes the soundness error γ^u that is negligible for large enough u. As a result, a recipient receives a total of ωu tokens.

3 Password-authenticated Delegation

We define the password-authenticated delegation (PAD) model, and show secure constructions realizing this model in the general case (i.e., for any cryptographic functionality) and more efficient customized ones for decryption and signatures.

3.1 Definition

We define an ideal functionality $\mathcal{F}_{PAD}$ for PAD of cryptographic capabilities. Our notion is a variation of the $\mathcal{F}_{BE}$ notion from [5]; in particular, we add the

Functionality $\mathcal{F}_{PAD}$

$\mathcal{F}_{PAD}$ is parameterized by a security parameter κ, a circuit class $\mathcal{C}_\kappa$, and a positive integer n.

Delegate: Upon receiving the command $(\mathsf{Delegate}, P_2, C, p)$ from party P_1 (the delegator), where P_2 is the delegatee, $C \in \mathcal{C}_\kappa$, and p is a password, if this is not the first activation, then do nothing. Otherwise:

- Send $(\mathsf{Delegate}, P_1, P_2)$ to the adversary.
- Upon receiving (OK) from the adversary, store $(C, p, j = 0, \mathsf{hflag} = 1)$, and output $(\mathsf{Delegate}, P_1)$ to P_2.

Evaluate: Upon receiving input $(\mathsf{Evaluate}, p', x)$ from P_2, where $x \in \{0,1\}^*$: if no stored state exists, end activation. Else, retrieve $(C, p, j, \mathsf{hflag})$, if $j > 0$, then end activation. Otherwise, increment j, and if $p' = p$ output $(C(x))$ to P_2.

Corrupt-evaluate: Upon receiving the command $(\mathsf{Corrupt\text{-}evaluate}, p', x)$ from the adversary, if no stored state exists, end activation. Else:

- Retrieve $(C, p, j, \mathsf{hflag})$.
- If $\mathsf{hflag} = 1$ and $j > 0$, or $j = n$, then end activation. Else, increment j, set $\mathsf{hflag} = 0$, and if $p' = p$ send $(C(x))$ to the adversary.

Fig. 1: An ideal functionality for PAD of cryptographic capabilities.

password authentication part. Our goal is to permit a party P_1 (the delegator) to delegate a cryptographic capability to party P_2 (the delegatee). These two parties share a password p drawn from a password space $\mathcal{P}$. P_2 will be able to execute the delegated functionality for only a limited number of times. For simplicity, we present a model allowing the honest recipient (and semi-honest one) for one evaluate query. This can be easily extended to multiple queries.[1]

Adversary model. We distinguish two types of adversaries: outsider who does not control P_2, and insider who controls P_2. An outsider adversary does not know the password, so all that she can do is trying n password guesses (such that if the i^{th} password guess is correct, she can execute the capability for additional $n - i$ times). An insider adversary knows the password and will execute the capability one time (if semi-honest) or up to n times (if malicious).

Ideal functionality. Our ideal functionality $\mathcal{F}_{PAD}$ is shown in Figure 1. It supports three interfaces. The first is $\mathsf{Delegate}$ that allows P_1 to delegate a capability, represented by the circuit C, to P_2. If the adversary agrees to continue, $\mathcal{F}_{PAD}$ creates a state and notifies P_2 about the delegation. The state stores the password p needed to authenticate an evaluation request, and a counter to track the number of evaluate queries performed so far, which is initialized to 0. It also stores a flag, hflag, tracking whether P_2 is honest or corrupt—this is needed

[1] In terms of the construction, this can be done by modifying $\mathcal{F}_{CT}$ to allow for t queries (for a small integer t) and so a malicious adversary will have tn queries.

since an honest gets one evaluate query while a corrupt (malicious) gets up to n queries. This flag is set by default to 1 indicating that the recipient is honest.

The second interface, Evaluate, allows P_2 to request evaluating the delegated capability over an input x. P_2 authenticates her request by showing a password p'. $\mathcal{F}_{PAD}$ outputs $C(x)$ only if p' matches the stored password p. $\mathcal{F}_{PAD}$ increments the counter j, and thus, all future queries will not output anything since an honest (or semi-honest) P_2 gets only one query.

The third interface Corrupt-evaluate is used by a malicious adversary. It notifies $\mathcal{F}_{PAD}$ that the environment wants to corrupt P_2, so it results in setting hflag $= 0$. Corrupt-evaluate allows the adversary to submit up to n evaluate queries. After these n invocations, which are tracked using the counter j, $\mathcal{F}_{PAD}$ will stop responding. A malicious outsider adversary, who does not know p, is basically trying to guess the correct password with each query she submits. A malicious insider adversary, on the other hand, knows p and can request executing C over up to n inputs. As shown, a delegated capability can be in the hand of either an honest/semi-honest P_2 or the malicious adversary, but not both at the same time. In particular, if an honest/semi-honest party performs their single evaluate query, Corrupt-evaluate will not do anything.[2]

Remark 1 (Honest P_1). $\mathcal{F}_{PAD}$ allows corrupting only P_2, so it is assumed that the delegator P_1 is honest. This makes sense for delegation as the goal is to permit someone else (the delegatee) to perform a particular functionality on behalf of the delegator. Nonetheless, $\mathcal{F}_{PAD}$ can be extended to allow corrupting P_1, and any construction realizing $\mathcal{F}_{PAD}$ must provide guarantees that P_1 provides the agreed-upon functionality for delegation.

Remark 2 (Anonymous delegation). $\mathcal{F}_{PAD}$ can be modified to support anonymous delegation so that the identity of the delegatee is not revealed. This can be achieved using the following edits: (1) P_1 does not disclose the identity of P_2 when requesting a delegation, (2) $\mathcal{F}_{PAD}$ notifies the adversary about a delegation request without disclosing the identity of P_2, and (3) it is P_1's responsibility to notify P_2 about the delegation in a private way. Note that both evaluate interfaces do not require the identity of P_2, thus the modified $\mathcal{F}_{PAD}$ preserves delegation anonymity.

3.2 Generic Construction for a PAD Scheme

We show a construction that securely realizes $\mathcal{F}_{PAD}$ for any function, which resembles a password-authenticated $(1, n)$-time program. Note that the difference is that in $(1, n)$-time programs, the set of keys used with each of the ω tokens is public. In our PAD construction, this set is secret and can be computed only by a party who knows the password p. The scheme is based on a simple idea; generate the token key set K (used with each of the ω tokens) using random

[2] Corrupt-evaluate could be split into two interfaces: one signals that the environment wants to corrupt P_2, and another permits a corrupt P_2 to request up to n evaluations. To simplify $\mathcal{F}_{PAD}$, we chose to have one interface encapsulating both.

Protocol 1 (A password authenticated $(1, n)$-time program scheme)

For a security parameter κ, positive integer n, input space $\mathcal{X}$, token key space $\mathcal{K}$, and password space $\mathcal{P}$, let P_1 be the delegator, P_2 be the delegatee, $[\omega, d, \delta]_q$ be a linear code C with alphabet size q, dimension d, distance δ, codeword size ω, and a generating matrix G such that $|C| = |\mathcal{X}|$. Also, let $g_0, \ldots, g_{q-1}$ be as described earlier, $\mathcal{F}_{BPO}$ be the ideal functionality of bounded-query digital lockers, and $(1, n)$-Prog be as defined in Figure 6 (Appendix A). Construct a tuple of algorithms (Delegate, Evaluate) for a PAD scheme as follows.

Delegate: On input a password $p \in \mathcal{P}$ and a function f with input space $\mathcal{X}$, P_1 does the following:

1. Generate a random seed $s \leftarrow \{0,1\}^\kappa$.
2. Generate a digital locker for the point function $I_{p,s}$ by sending the command (Obfuscate, P_2, p, s) to $\mathcal{F}_{BPO}$ ($\mathcal{F}_{BPO}$ will notify P_2 about the obfuscation).
3. Generate a set of token keys $K = \{k_0, \ldots, k_{q-1}\}$: for $i \in \{0, \ldots, q-1\}$, generate a pseudorandom string $r_i = PRG(s)[i]$ and a token key $k_i = g_i(r_i)$.
4. Obtain $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\omega, eP) = (1, n)\text{-Prog.Encap}(f, \omega, q, G, K)$ and send $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\omega, eP)$ to P_2.

Evaluate: On input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\omega, eP)$, $x \in \mathcal{X}$, and $p \in \mathcal{P}$, P_2 does the following:

1. Retrieve the seed s by sending the command (Evaluate, p) to $\mathcal{F}_{BPO}$.
2. Generate the token key set K as above.
3. Output $f(x) = (1, n)\text{-Prog.Eval}(\mathsf{ct}_1, \ldots, \mathsf{ct}_\omega, eP, x, K)$.

Fig. 2: A construction of a PAD scheme for general functionalities.

mapping over a pseudorandom string, generated using a prseudorandom number generator (PRG), while storing the PRG seed s in a digital locker. That is, $PRG(s)$ outputs a string of length $q\kappa$ bits—so it is composed of q substrings each of length κ and we refer to the i^{th} substring as $PRG(s)[i]$. We denote these substrings as $r_0, \ldots, r_{q-1}$, where q is the size of the code alphabet used in $(1, n)$-time programs. These substrings are mapped to token keys, where we define mapping functions $g_0, \ldots, g_{q-1}$, such that $g_i : \{0,1\}^\kappa \to \mathcal{K}$ for $i \in \{0, \ldots, q-1\}$.

Our PAD construction is shown in Figure 2. It uses the ideal functionalities $\mathcal{F}_{BPO}$ and $\mathcal{F}_{CT}$ from [5] (the latter is used in the $(1, n)$-time program protocol). To accommodate the change in token key generation, we modify the $(1, n)$-time program protocol from [5] to provide the set of keys K as input instead of being generated at random within the protocol itself (for completeness, we provide the modified version in Appendix A with the changes highlighted in red). In Figure 2, we invoke the Encap and Eval procedures from this protocol while passing the token key set K as one of the inputs.[3]

[3]One may argue that it suffices to use $\mathcal{F}_{BE}$ in our construction and prove security in the ($\mathcal{F}_{BPO}$, $\mathcal{F}_{BE}$)-hybrid model instead of invoking the actual $(1, n)$-time program protocol in the ($\mathcal{F}_{CT}$, $\mathcal{F}_{BPO}$)-hybrid model. However, configuring the token keys is a

Our construction provides the guarantee that an honest/semi-honest party who knows p can execute the delegated capability once, an insider malicious adversary (who knows p) can execute it up to n times, and an outsider malicious adversary (who does not know p) can make at maximum n password guesses. If this adversary manages to guess the password correctly, she can also exercise the delegation for up to n times. So this is a weaker guarantee than what $\mathcal{F}_{PAD}$ offers (i.e., guessing p successfully in the i^{th} query allows for additional $n - i$ evaluations rather than n).

Security. Our construction securely realizes $\mathcal{F}_{PAD}$ (with the exception of the weaker guarantee stated above) in the ($\mathcal{F}_{CT}$, $\mathcal{F}_{BPO}$)-hybrid model and assuming one-way functions as we use a PRG in our construction (and sup-exponentially secure $i\mathcal{O}$ needed for $(1, n)$-time programs). The proof of the following theorem is straightforward; basically it relies on the security of PRGs and the $(1, n)$-time program scheme (the full proof of the latter can be found in [5]). The only addition is a hybrid in which the randomly generated key set K in the original $(1, n)$-time program scheme is replaced with pseudorandomly generated one (using the PRG idea), which is indistinguishable by the security of PRGs. Based on that, we have the following theorem.

Theorem 1. *Assuming one-way functions and sup-exponentially secure $i\mathcal{O}$, the construction described in Figure 2 realizes $\mathcal{F}_{PAD}$, but with n evaluations upon a successful adversarial password guess, in the ($\mathcal{F}_{CT}$, $\mathcal{F}_{BPO}$)-hybrid model.*

3.3 Customized PAD Schemes

The previous construction, although powerful as it enables constructing a PAD of any cryptographic functionality, requires a strong assumption—$i\mathcal{O}$. In this section, we present alternative constructions for particular functionalities that avoid $i\mathcal{O}$ but with additional setup and/or alternative (weaker) assumptions.

Bulletin board-based schemes. Here we assume a public bulletin board that resembles an append-only log accessible by all parties (which could be instantiated in a decentralized way using a blockchain, for example). We show how this could be useful for instantiating a PAD scheme for decryption and digital signatures. The two parties P_1 and P_2 share a password p, and both have access to the bulletin board. Having a bulletin board allows the following: P_1 posts encrypted information on the board that P_2 can access only if she has the key that allows decrypting this information.

PAD for decryption. P_1 simply posts ciphertext c on the board that is encrypted using a symmetric key sk. Then she creates a bounded-query digital locker (through $\mathcal{F}_{BPO}$) storing $m = \mathsf{sk}$. P_2 who knows the correct password p can retrieve sk and decrypt c.

PAD for digital signatures. This is an extension of the decryption construction. Assume we have a tokenizable digital signature scheme that allows generating one-time signing tokens (i.e., each signing token can be used to sign only one

protocol detail (it does not appear in $\mathcal{F}_{BE}$), for this reason we use the protocol directly (which is in the $\mathcal{F}_{CT}$-hybrid model).

message). The PAD signature construction proceeds as follows: P_1 generates t tokens, encrypts them and posts the ciphertext c on the board, and then creates a bounded-query digital locker for P_2 that stores the decryption key sk as above.

Both constructions are secure against malicious insider and outsider adversaries. That is, the consumable token power gap is not useful; having additional queries for the malicious adversary (insider who knows p or an outsider who manages to guess p) will not help since there is only one piece of information stored in the locker, i.e., sk, and this adversary (just like an honest party) can use it to decrypt one ciphertext (and then sign only t message in case of signatures).

A concrete construction based on this idea can be instantiated using the tokenizable digital signature scheme from [7]. In a nutshell, we have P_1 who owns a keypair (x, g^x) (where g is the generator for a cyclic group $\mathbb{G}$ of a prime order q and $x \in \mathbb{Z}_q$). The scheme relies on producing two-layered Schnorr signature: the first (which is the signing token z) is a Schnorr signature over a random element $k \in \mathbb{Z}_q$, while the second layer is a Schnorr signature over the message m using z as the signing key. Verifying the signature will be against g^x and it verifies both layers at the same time. The one-time property is enforced using the bulletin board; any accepted valid signature will have its unique value k recorded on the board, and consequently any signature with an already-published k value will be rejected. This will prevent using the token for signing multiple messages.[4]

Without a bulletin board. Here, instead of posting the ciphertext on the board, it will be sent directly to P_2. However, for signatures, the construction based on [7] will not work anymore; the lack of having a bulletin board will allow reusing a token to sign as many messages as P_2 wishes. This is fine if the bounded-execution feature is not required; in other words, think of delegation in which P_1 issues a delegation key to P_2 that enables signing any number of messages. So one token is enough which is viewed as a delegation key.

Remark 3. The encryption-based PAD signature constructions above represent a framework in the sense that any non-interactive proxy or delegatable signature scheme can be used. The original signer P_1 produces the delegation information, encrypts them, and sends a digital locker containing sk (the ciphertext is either sent directly or posted on the board). The guarantee on the delegation, whether one-time or reusable, depends on the guarantees offered by the underlying proxy or delegatable signature scheme.

Hash equivocation-based construction. An alternative PAD signature construction can be based on Chameleon hashing as proposed in [6] that works for a small message space $\mathcal{M}$ such that $|\mathcal{M}| = v$. P_1 signs a dummy message using the hash-then-sign paradigm, then she computes the equivocation of the

[4] The work [7] introduces techniques to support timed delegation (so a delegation can be exercised within a preset timeframe), revocability (allowing P_1 to revoke delegation), and enforcing a policy restricting which message class can be signed. We focus on the one-time tokenizable aspect, while we note that their techniques can be used to support these additional properties in the PAD setting. Also, since this construction supports delegation anonymity, our PAD construction will enjoy this feature as well.

hash of this dummy message for all messages in $\mathcal{M}$. These equivocations are stored in a multi-message bounded-query digital locker. The disadvantage here is the impact of the consumable token power gap. An insider malicious adversary who knows p can sign up to n messages (or an outsider who manages to guess the password correctly can sign additional $n-i$ message if he succeeds in the i^{th} guess, but still this respects the guarantee provided by $\mathcal{F}_{PAD}$), while an honest P_2 can sign only one message.[5]

The multi-message bounded-query digital locker can be constructed as follows. Each of the messages $m_1, \ldots, m_v$ is shared into u shares. The first share of these messages will be stored in a consumable token under keys generated using random mappings of $p \parallel j$ for $j \in \{1, \ldots, v\}$. Then for each message, the rest of its shares are stored in $u-1$ tokens chained in the same way as in bounded-query digital lockers. This will amplify the soundness error of a consumable token into negligible while preserving the number of queries (or password guesses) to be n (follows using the security proof of digital lockers from [5]).

Security. Security relies on the security of the symmetric encryption scheme and bounded-query digital lockers, and in case of requiring a bulletin board, the security of this board (satisfying liveness and safety [23]). The hash equivocation-based scheme relies on the security of the multi-message bounded-query digital lockers described above and the correctness and collision resistance of Chameleon hash functions (as defined in [9]).

4 Password-authenticated MPC

We present the password-authenticated MPC (PAMPC) model and show secure constructions realizing this notion for both the interactive and non-interactive case under various adversary models.

4.1 Definition

We define an ideal functionality $\mathcal{F}_{PAMPC}$ for the PAMPC model. We adopt the same adversary model as in Section 3 with respect to insider and outsider adversaries who could be semi-honest or malicious. In a sense, this model represents a notion of MPC among friends; only those who share the same password can participate. That is, for any function $f(x_1, \ldots, x_w)$ and password p, where w is the number of parties providing private inputs $x_1, \ldots, x_w$, evaluating f over these inputs cannot take place unless all these parties provide the password p.

Our ideal functionality $\mathcal{F}_{PAMPC}$ is shown in Figure 3. It supports two interfaces. The first is Compute that allows P_i to supplement her private input x_i and the password p_i that she knows. If the adversary agrees to continue, $\mathcal{F}_{PAMPC}$ creates a state for P_i recording her inputs x_i and p_i, and indicating that this

[5]Indeed, an alternative approach is to store signatures over the messages in $\mathcal{M}$ instead of the equivocated hashes. However, a consumable token can store short strings, and so storing hashes is more viable than storing signatures that could be much longer.

Functionality $\mathcal{F}_{PAMPC}$

$\mathcal{F}_{PAMPC}$ is parameterized by a security parameter κ, a positive integer n. Upon initiation, a counter ctr and a compute flag cflag are initialized to 0, and $\mathcal{F}_{PAMPC}$ is supplied with a password $p \in \mathcal{P}$ and function $f : \{\{0,1\}^*\}^w \to \{0,1\}^*$, where $\mathcal{P}$ is the password space and w is a positive integer.

Compute: Upon receiving the command $(\mathsf{Compute}, P_i, x_i, p_i)$ from party P_i, where $x_i \in \{0,1\}^*$ and p_i is a password, if this is not the first activation from P_i, then do nothing. Otherwise:

- Send $(\mathsf{Compute}, P_i)$ to the adversary.
- Upon receiving (OK) from the adversary, store $(P_i, x_i, p_i, j = 1, \mathsf{hflag}_i = 1)$ and increment ctr by 1.
- If $\mathsf{ctr} = w$, then if $p_i = p$ for all $i \in \{1, \ldots, w\}$ and $\mathsf{cflag} = 0$, set $\mathsf{cflag} = 1$ and output $f(x_1, \ldots, x_w)$ to $P_1, \ldots, P_w$, else, do nothing.

Corrupt-compute: Upon receiving the command $(\mathsf{Corrupt\text{-}compute}, P_i, x_i, p_i)$ from the adversary, if there is a state stored for P_i, retrieve $(P_i, x_i, p_i, j, \mathsf{hflag}_i)$, else create state $(P_i, \perp, \perp, j = 0, \mathsf{hflag}_i = 0)$. If $\mathsf{hflag}_i = 1$ then end activation, else:

- If $j = n$, then end activation. Else, increment ctr if $j = 0$, increment j and update the state of P_i with x_i and p_i.
- If $\mathsf{ctr} = w$, then if $p_i = p$ for all $i \in \{1, \ldots, w\}$ and $\mathsf{cflag} = 0$, then set $\mathsf{cflag} = 1$ and output $f(x_1, \ldots, x_w)$ to $P_1, \ldots, P_w$, else do nothing.

Fig. 3: An ideal functionality for PAMPC.

party is honest by setting the flag hflag_i to 1. It also stores a query counter for this party that is set to 1 upon the first activation. As shown, an honest party can perform only one compute query. $\mathcal{F}_{PAMPC}$ tracks the number of inputs supplemented so far using the counter ctr.

The second interface, $\mathsf{Corrupt\text{-}compute}$, is used by the adversary. As before, it notifies $\mathcal{F}_{PAMPC}$ that the environment wants to corrupt P_i and provides inputs x_i and p_i for this corrupt party. $\mathsf{Corrupt\text{-}compute}$ allows the adversary to submit up to n compute queries (per corrupt party). After these n invocations, which are tracked using the counter j, $\mathcal{F}_{PAMPC}$ will stop responding (so no more password guesses can be made by the corrupt P_i). As shown, corrupting an honest party who already performed one compute query will not provide the adversary with any benefit; this party has already exhausted her one query. Furthermore, the counter ctr is incremented only once for any corrupt party even if it invokes $\mathsf{Corrupt\text{-}compute}$ multiple times.

After each $\mathsf{Compute}/\mathsf{Corrupt\text{-}compute}$ query, $\mathcal{F}_{PAMPC}$ checks if $\mathsf{ctr} = w$, and if so, it proceeds with verifying that all parties provided the correct password p. If this is the case, $\mathcal{F}_{PAMPC}$ outputs the evaluation of f over the parties' inputs.[6]

[6] $\mathcal{F}_{PAMPC}$ covers the fairness property as the computed output is sent to all parties. Satisfying this property depends on the underlying MPC protocol in our constructions; the one in Section 4.2 does not satisfy fairness, while the one in Section 4.3 achieves

As noted, $\mathcal{F}_{PAMPC}$ allows computing the function f only once (tracked using the flag cflag). Thus, it prevents leaking the residual function.[7]

4.2 Two-party Password-authenticated Non-interactive MPC

We show a construction for two-party non-interactive MPC (NIMPC) that combines garbled circuits with a password-authenticated non-interactive oblivious transfer (OT) functionality. Recall that in garbled circuits, P_1 (the garbler) garbles the circuit and sends it along with her input labels to P_2 (the evaluator). Then, P_1 engages with P_2 in an interactive OT protocol so that P_2 can obtain the wire labels corresponding to her input. P_2 uses these labels to evaluate the circuit and obtain the output, after which she shares the output with P_1. We transform this construction into a password-authenticated and non-interactive one. In particular, we introduce a modified version of bounded-query digital lockers that resembles a non-interactive OT scheme. Instead of storing one message in the locker, we store two messages at two valid points (representing the labels of 0 and 1 for a wire). The two points, or token keys, are derived based on the password, namely, based on $p \parallel 0$ and $p \parallel 1$, respectively, thus adding the password authentication feature. Clearly, without knowing the correct password, P_2 cannot retrieve the labels and will not be able to evaluate the circuit.

However, due to the power gap between the honest party and the adversary in the consumable token model, this construction is secure against a semi-honest insider adversary and a malicious outsider adversary. That is, if a malicious adversary corrupts P_2, he gets a hold on the password p and can retrieve both wire labels since this adversary can perform n queries (instead of one as for the honest/semi-honest P_2). A malicious outsider adversary can try up to n password guesses. Naturally, if this adversary gets lucky and guesses the password in any of the first $n-1$ queries, he will be able to obtain both wire labels, and if he guesses the password in his n^{th} query, he can evaluate the circuit just like P_2. So for a malicious outsider adversary the only viable attack is trying n password guesses, with probability $\frac{n-1}{|\mathcal{P}|}$ he compromises the protocol security, while with probability $\frac{n}{|\mathcal{P}|}$ he can evaluate although he is not the legitimate party.[8]

Password-authenticated non-interactive OT. Password-authenticated OT cannot be realized using the conventional OT ideal functionality, i.e., P_1 provides two strings l^0 and l^1, and P_2 provides a bit b and the output is l_b sent to P_2 (and so for input of length ℓ bits, this functionality is initiated ℓ times). In

fairness in the honest majority setting, otherwise additional techniques are needed to have fairness.

[7] The residual function is computing f over the honest parties' inputs and all combinations of adversarial inputs, a notion that has been introduced in non-interactive MPC [26]. In our case, an adversary computes the function f over the honest parties inputs, and up to n^t combinations of the inputs of t corrupt parties.

[8] For simplicity, these probabilities consider a uniform password distribution. For an arbitrary password distribution, the probability will be the number of queries multiplied by the password guess probability based on the underlying distribution.

Functionality $\mathcal{F}_{PAOT}$

$\mathcal{F}_{PAOT}$ is parameterized by a security parameter κ and a positive integer n.

Send: Upon receiving the command $(\mathsf{Send}, P_2, p, (x_1^0, x_1^1), \ldots, (x_\ell^0, x_\ell^1))$ from the sender P_1, where $x_i^b \in \{0,1\}^\kappa$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0,1\}$) and p is a password, if this is not the first activation from P_1, then do nothing. Otherwise:

- Send $(\mathsf{Send}, P_1, P_2)$ to the adversary.
- Upon receiving (OK) from the adversary, store $(p, (x_1^0, x_1^1), \ldots, (x_\ell^0, x_\ell^1), j = 0, \mathsf{hflag} = 1)$ and output (Send, P_1) to P_2.

Receive: Upon receiving the command $(\mathsf{Receive}, p', b_1, \ldots, b_\ell)$ from the receiver P_2, if there is no state stored, then do nothing. Else, retrieve $(p, (x_1^0, x_1^1), \ldots, (x_\ell^0, x_\ell^1), j, \mathsf{hflag})$, if $j > 0$, then do nothing. Else, increment j, then if $p' = p$, send $x_1^{b_1}, \ldots, x_\ell^{b_\ell}$ to P_2, otherwise do nothing.

Corrupt-receive: Upon receiving the command $(\mathsf{Corrupt\text{-}receive}, p', b_1, \ldots, b_\ell)$ from the adversary, if there is no state stored, then do nothing. Else, retrieve $(p, (x_1^0, x_1^1), \ldots, (x_\ell^0, x_\ell^1), j, \mathsf{hflag})$, if $\mathsf{hflag} = 1$ and $j > 0$, or $j = n$, then do nothing. Else, do the following:

- Increment j and set $\mathsf{hflag} = 0$.
- If $p' = p$, then send $x_1^{b_1}, \ldots, x_\ell^{b_\ell}$ to the adversary, otherwise, do nothing.

Fig. 4: An ideal functionality for password-authenticated OT.

the password-authenticated setting (and especially under the consumable token model), OT of all ℓ wires are connected with each other via the password p (i.e., all are queried based on one password). If independent OT instantiations are used, such that all use the same p, this amplifies the number of password guesses the adversary will have (a total of $n\ell$ guesses). Thus, what we need is an OT functionality that allows P_1 to input ℓ pairs of strings $\{(l_1^0, l_1^1), \ldots, (l_\ell^0, l_\ell^1)\}$, and allows P_2 to input selection bits $\{b_1, \ldots, b_\ell\}$ and retrieve $\{l_1^{b_1}, \ldots, l_\ell^{b_\ell}\}$ in return only if the correct password is presented.[9]

Ideal functionality. We define an ideal functionality for the above in Figure 4 denoted as $\mathcal{F}_{PAOT}$. As shown, $\mathcal{F}_{PAOT}$ allows the sender P_1 to specify the set of ℓ pairs of messages and the password p guarding the retrieval of these messages. It also enables the receiver who knows the correct password p to retrieve ℓ messages composed of one message from each of the ℓ pairs that P_1 sent. $\mathcal{F}_{PAOT}$ allows the malicious adversary to try up to n password guesses. As noted, since we define $\mathcal{F}_{PAOT}$ in light of the consumable token model, if this (outsider) malicious adversary succeeds in guessing the correct password in the first $n-1$ queries, he can retrieve all ℓ message pairs (i.e., the guess will be made with all $b_i = 0$, then after guessing the correct p, he can make a query with all $b_i = 1$).

[9] It should be noted that a form of password-authenticated oblivious transfer has been shown in [11] under the notion of Oblivious Language-Based Envelope (OLBE), but it is an interactive protocol.

Protocol 2 (A password-authenticated non-interactive OT scheme)

For a security parameter κ, positive integers u and ℓ, a password space $\mathcal{P}$, and a token key space $\mathcal{K}$, let $g_1, \ldots, g_{u\ell}$ be $g_1 : \mathcal{P} \times \mathbb{Z} \to \mathcal{K}$ and $g_i : \mathcal{P} \times \mathbb{Z} \times \{0,1\}^\kappa \to \mathcal{K}$ for $i > 1$, P_1 be the sender, P_2 be the receiver, and $\mathcal{F}_{CT}$ be the consumable token functionality. Construct a tuple of algorithms $(\mathsf{Send}, \mathsf{Receive})$ as follows.

Send: on input labels $\{(l_1^0, l_1^1), \ldots, (l_\ell^0, l_\ell^1)\}$ and password p, P_1 does the following:

1. Generate u random shares for each label l_i^b for $b \in \{0,1\}$ and $i \in \{1, \ldots, \ell\}$ as $l_{i,1}^b, \ldots, l_{i,u}^b$ such that $l_i^b = \oplus_{j=1}^{u} l_{i,j}^b$.
2. For $i \in \{1, \ldots, \ell\}$, set $r_0^i = r'_0 = \perp$ and generate ℓ sets of random strings $r_v^i \leftarrow \{0,1\}^\kappa$ for $v \in \{1, \ldots, u\}$. Also, generate another set of random strings $r'_v \leftarrow \{0,1\}^\kappa$ for $v \in \{1, \ldots, \ell\}$.
3. For $i \in \{1, \ldots, \ell\}$:
   (a) Compute $s_i = \oplus_{j=0}^{i-1} r'_j$.
   (b) Generate two token keys $k_b \leftarrow g_{(i-1)u+1}(p \,\|\, b, s_i)$ for $b \in \{0,1\}$.
   (c) Generate a token $\mathsf{ct}_{(i-1)u+1}$, with a unique token ID $\mathsf{tid}_{(i-1)u+1}$, encoding m_0 and m_1 using k_0 and k_1, respectively, by sending the command $(\mathsf{Encode}, \mathsf{tid}_{(i-1)u+1}, \{k_0, k_1\}, \{m_0, m_1\}, 2)$ to $\mathcal{F}_{CT}$, such that $m_b = l_{i,1}^b \,\|\, r_1^i \,\|\, r'_i$.
   (d) For $w \in \{2, \ldots, u\}$:
      i. Compute $s_w = \oplus_{j=0}^{w-1} r_j^i$.
      ii. Generate two token keys $k_b \leftarrow g_{(i-1)u+w}(p \,\|\, b, s_i)$ for $b \in \{0,1\}$.
      iii. Generate a token $\mathsf{ct}_{(i-1)u+w}$, with a unique token ID $\mathsf{tid}_{(i-1)u+w}$, encoding m_0 and m_1 using k_0 and k_1, respectively, by sending the command $(\mathsf{Encode}, \mathsf{tid}_{(i-1)u+w}, \{k_0, k_1\}, \{m_0, m_1\}, 2)$ to $\mathcal{F}_{CT}$, such that $m_b = l_{i,w}^b \,\|\, r_w^i$.

$\mathsf{Receive}$: on input x of length ℓ bits, $u\ell$ tokens $\mathsf{ct}_1, \ldots, \mathsf{ct}_{u\ell}$, and password $p \in \mathcal{P}$, P_2 does the following:

1. For $i \in \{1, \ldots, \ell\}$ set $r_0^i = r'_0 = \perp$.
2. For $i \in \{1, \ldots, \ell\}$ do the following:
   (a) Compute $s_i = \oplus_{j=0}^{i-1} r'_j$.
   (b) Generate a token key $k \leftarrow g_{(i-1)u+1}(p \,\|\, x_i, s_i)$.
   (c) Query token $\mathsf{ct}_{(i-1)u+1}$ using k to retrieve $m = l_{i,1}^{x_i} \,\|\, r_1^i \,\|\, r'_i$ by sending the command $(\mathsf{Decode}, \mathsf{tid}_{i+u}, k)$ to $\mathcal{F}_{CT}$.
   (d) for $w \in \{2, \ldots, u\}$:
      i. Compute $s_w = \oplus_{j=0}^{w-1} r_j^i$.
      ii. Generate a token key $k \leftarrow g_{(i-1)u+w}(p \,\|\, x_i, s_i)$
      iii. Query token $\mathsf{ct}_{(i-1)u+w}$ using k to retrieve m by sending the command $(\mathsf{Decode}, \mathsf{tid}_{(i-1)u+w}, k)$ to $\mathcal{F}_{CT}$, where $m = l_{i,w}^{x_i} \,\|\, r_w^i$.
   (e) Compute $l_i^{x_i} = \oplus_{j=1}^{u} l_{i,j}^{x_i}$
3. Output $l_1^{x_1}, \ldots, l_\ell^{x_\ell}$

Fig. 5: A construction for password-authenticated non-interactive OT scheme.

Construction. Our construction is shown in Figure 5. It is based on the idea of having a set of ℓ two-message digital lockers such that the i^{th} locker is com-

posed of u tokens each of which is storing a share pair of the i^{th} wire labels (and all these u tokens are chained together through the randomness needed to generate the token keys). Furthermore, the ℓ lockers are chained together (via the first token storing the first share pair of the i^{th} wire labels) also through the randomness. As such, P_2 is forced to work on the first token first, then she can work sequentially on retrieving the first share of each wire label, and just at that time, she can retrieve the rest of the u shares of each wire label (also working sequentially on the token storing the $u-1$ shares of each wire label). The chaining is needed to avoid amplifying the number of password guesses an adversary obtains, and the sharing is needed to amplify the consumable token non-negligible soundness error to negligible. Without this randomness, P_2 cannot generate the key corresponding to her selection bits (which are the bits in her input x in the figure).

The honest (or semi-honest) recipient can retrieve only one label share from each token. A malicious adversary, will have negligible advantage over $\frac{n}{|\mathcal{P}|}$ to guess the correct password despite sending multiple tokens. That is, this adversary can make n password guesses (with a success probability of $\frac{n}{|\mathcal{P}|}$) only over the first token, for the rest he has to guess the random string used to map the password to a token key as well thus reducing the probability to negligible.

Security. We show security of the OT construction and the two-party PAN-IMPC protocol that uses this OT functionality. The following theorem follows using the same proof of bounded-query digital lockers from [5].

Theorem 2. *For $0 \leq \gamma \leq 1$, if each of $g_1, \ldots, g_{u\ell}$ is as defined in Figure 5, then for large enough positive integer u the protocol in Figure 5 securely realizes $\mathcal{F}_{PAOT}$ in the $\mathcal{F}_{CT}$-hybrid model.*

For the NIMPC protocol, it proceeds as follows. P_1 garbles the circuit, prepare OT tokens for P_2's labels using the construction in Figure 5, and sends the circuit, the tokens, and the labels of her input to P_2. P_2 retrieves the labels of her input from the tokens using p, then evaluates the circuit and notifies P_1 about the output. It is easy to see that the security of this construction (which is in the $\mathcal{F}_{PAOT}$-hybrid model) relies on the security of the OT scheme and the security of garbled circuits, and it is tied to the success probability of an outsider malicious adversary discussed before.[10] Based on that we have the following theorem:

Theorem 3. *The above NIMPC protocol realizes $\mathcal{F}_{PAMPC}$ for two parties in the $\mathcal{F}_{PAOT}$-hybrid model against semi-honest adversaries, and against malicious outsider adversaries with probability $1 - \frac{n}{|\mathcal{P}|}$.*

Remark 4. Note that the generic construction for PAD in Section 2 is also a two party NIMPC; P_1 will include her input x in the obfuscated program of $f(x, y)$. It is a stronger one as it only reveals the residual function for n adversarial inputs

[10] Defending against a malicious garbler can be done using existing methods in the literature on garbled circuits, the only difference is that instead of interactive OT, we have the non-interactive consumable token-based one.

in case of a malicious P_2 rather than compromising security as discussed earlier. However, the downside is that it requires $i\mathcal{O}$.

4.3 Password-authenticated Interactive MPC

The basic idea is to utilize the secret sharing-based MPC paradigm. Take two parties as an example, P_1 has private input x and P_2 has private input y and they want to compute $f(x, y)$. P_1 shares her input as x_1 and x_2, and P_2 does the same to obtain y_1 and y_2. P_1 then sends a bounded-query digital locker storing x_2 to P_2 such that P_2 will be able to access x_2 if she knows the password p, and P_2 does the same to send the share y_2 to P_1. Once both parties have the shares they can perform the MPC protocol over the shares.

Hence, without knowing the password p, these parties cannot access the shares, and so cannot perform the MPC protocol. Furthermore, this scheme is secure against a malicious (insider and outsider) adversaries since the locker stores one piece of information and the additional data retrieval queries a malicious adversary may obtain are not useful. If the same password p is used to secure tokens containing x_2 and y_2, the number of password trials the (outsider) adversary obtains is $2n$ instead of n. That is, she uses the first token (from P_1 to P_2) to try $n-1$ passwords, if none is successful, then she uses the second token (from P_2 to P_1) to make another n password guesses. If one is successful, this reveals y_2 and the correct password, so she can go back to the first token and obtain x_2 as well. Nonetheless, this does not compromise input privacy since all what this adversary obtains are random shares that do not leak any information about the input. Moreover, in case of an outsider adversary (i.e., man in the middle attacker), since the (physical) tokens did not reach their destination these parties will know that the tokens have been intercepted and will abort the protocol (and change the password).[11]

The above can be extended to multiple parties using the same logic. Say we have w parties, now each party P_i sends $w-1$ digital lockers (one for every other player containing a share of her input). Security relies on the security of digital lockers and the security of the (secret sharing-based) MPC protocol.

5 Concluding Remarks and the Road Ahead

In this paper, we showed various constructions adding the password-authenticated feature to advanced cryptographic applications including delegation and MPC. All of these were done under the consumable token model, thus investigating the capabilities this model can offer and its limitations. The goals are introducing new notions for password-authenticated cryptographic primitives, and subsequently, igniting new directions to investigate other technologies and hardness

[11] This is not the case for MPC in the client-server model. A client sends all shares of her secret input to the servers (one share per server). So an adversary who intercepts the tokens and gets lucky in guessing the password, or it is an insider one who knows the password, will obtain all shares and reconstruct the private input.

assumptions to realize these notions (perhaps combined with the consumable token model to strengthen its capabilities).

Relying on $i\mathcal{O}$ provides a powerful tool. For example, the work in [24] which allows for a two-round MPC and guarantees that the protocol is executed only once, can be easily extended to support the password-authenticated feature. That is, the obfuscated program will have the password hardcoded, and then a party is required to present the password (inside the encryption of the opening of her input commitment). The obfuscated program will compute the output only if all parties know the correct password. Our MPC constructions do not require $i\mathcal{O}$ but use an alternative hardness assumption (i.e., unclonable polymers) and offer weaker security guarantees based on the limitations of this hardness assumption.

Our generic PAD scheme (Section 3.2), although requiring $i\mathcal{O}$, supports non-interactivity (only one round of communication is needed). This, however, comes at the expense of security of malicious (outsider) who obtains n password guesses, and even if he succeeds in the last guess he still can execute the delegated capability n times. An open question here is whether we can obtain a stronger security guarantee. That is, if the i^{th} password is successful, the adversary can execute the delegated capability for only the remaining queries, i.e., $n - i$. An idea here is to store the secret messages corresponding to the input codewords in a digital locker-like construction instead of the separate tokens we have now (and so there is no additional digital locker to store the PRG seed as we have now). However, extending the chaining-based digital locker construction to support that seems hard to do. What seems easy to do is to have each of the ω lockers be tied to a separate password (so the two parties shares ω passwords instead of one, thus breaking the connections between these lockers). We leave addressing the question of having only one password as a future work direction.

The protection that consumable tokens provide, as noted, relies on whether the adversary is an outsider or insider, and in most of our constructions security is achieved against semi-honest insider adversaries. This is inherited from the power gap between an honest recipient and the adversary where the latter can perform more than one query. At the same time, achieving bounded-execution in one communication round, i.e., the notion of one or k-time programs, is impossible without bounded-query memory devices even in the quantum model [14]. As pointed out in [6], based on the different features of the quantum and the unclonable polymer models, seems the path is to explore a hybrid model that combine these together to obtain the best of both worlds. Such a hybrid model has the potential of closing the power gap in consumable tokens, and thus, achieving the password-authenticated notion for stronger adversaries than what we achieve in this work. We also leave this as a future work direction.

References

1. Abdalla, M., Barbosa, M., Bradley, T., Jarecki, S., Katz, J., Xu, J.: Universally composable relaxed password authenticated key exchange. In: CRYPTO (2020)
2. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the j-pake password-authenticated key exchange protocol. In: IEEE S&P (2015)

3. Abdalla, M., Cornejo, M., Nitulescu, A., Pointcheval, D.: Robust password-protected secret sharing. In: ESORICS (2016)
4. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: CRYPTO (2022)
5. Almashaqbeh, G., Canetti, R., Erlich, Y., Gershoni, J., Malkin, T., Pe'er, I., Roitburd-Berman, A., Tromer, E.: Unclonable polymers and their cryptographic applications. In: EUROCRYPT (2022)
6. Almashaqbeh, G., Chatterjee, R.: Unclonable cryptography: A tale of two no-cloning paradigms. In: Secrypt (2023)
7. Almashaqbeh, G., Nitulescu, A.: Anonymous, timed and revocable proxy signatures. Cryptology ePrint Archive (2023)
8. Apon, D., Cachet, C., Fuller, B., Hall, P., Liu, F.H.: Nonmalleable digital lockers and robust fuzzy extractors in the plain model. In: ASIACRYPT (2022)
9. Ateniese, G., Magri, B., Venturi, D., Andrade, E.: Redactable blockchain–or–rewriting history in bitcoin and friends. In: IEEE EuroS&P (2017)
10. Bagherzandi, A., Jarecki, S., Saxena, N., Lu, Y.: Password-protected secret sharing. In: ACM CCS (2011)
11. Blazy, O., Chevalier, C., Germouty, P.: Adaptive oblivious transfer and generalization. In: ASIACRYPT (2016)
12. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: EUROCRYPT (2000)
13. Bradley, T., Camenisch, J., Jarecki, S., Lehmann, A., Neven, G., Xu, J.: Password-authenticated public-key encryption. In: ACNS (2019)
14. Broadbent, A., Gutoski, G., Stebila, D.: Quantum one-time programs. In: CRYPTO (2013)
15. Camenisch, J., Enderlein, R.R., Neven, G.: Two-server password-authenticated secret sharing uc-secure against transient corruptions. In: PKC (2015)
16. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: Virtual smart cards: How to sign with a password and a server. In: SCN (2016)
17. Camenisch, J., Lysyanskaya, A., Neven, G.: Practical yet universally composable two-server password-authenticated secret sharing. In: ACM CCS (2012)
18. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: EUROCRYPT (2008)
19. Chen, L., Huang, K., Manulis, M., Sekar, V.: Password-authenticated searchable encryption. International Journal of Information Security **20**(5), 675–693 (2021)
20. Das, P., Hesse, J., Lehmann, A.: Dpase: distributed password-authenticated symmetric-key encryption, or how to get many keys from one password. In: ASIA CCS (2022)
21. Di Raimondo, M., Gennaro, R.: Provably secure threshold password-authenticated key exchange. In: EUROCRYPT (2003)
22. Fenteany, P., Fuller, B.: Same point composable and nonmalleable obfuscated point functions. In: ACNS (2020)
23. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: EUROCRYPT (2015)
24. Garg, S., Polychroniadou, A.: Two-round adaptively secure mpc from indistinguishability obfuscation. In: TCC (2015)
25. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: CRYPTO (2008)
26. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: CRYPTO (2011)

27. Hao, F., Bag, S., Chen, L., van Oorschot, P.C.: Owl: An augmented password-authenticated key exchange scheme. Cryptology ePrint Archive (2023)
28. Hao, F., van Oorschot, P.C.: Sok: Password-authenticated key exchange–theory, practice, standardization and real-world lessons. In: AsiaCCS (2022)
29. Jarecki, S., Jubur, M., Krawczyk, H., Saxena, N., Shirvanian, M.: Two-factor password-authenticated key exchange with end-to-end security. ACM Transactions on Privacy and Security **24**(3), 1–37 (2021)
30. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and t-pake in the password-only model. In: ASIACRYPT (2014)
31. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: EUROCRYPT (2001)
32. MacKenzie, P., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on rsa. In: ASIACRYPT (2000)
33. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. In: CRYPTO (2002)
34. Roy, P.S., Dutta, S., Susilo, W., Safavi-Naini, R.: Password protected secret sharing from lattices. In: ACNS (2021)

A A Refined $(1, n)$-time Program Protocol

In this section, we show a modified version the original $(1, n)$-time program construction from [5] allowing to pass the token key set K as input (which we need in Section 3.2)—our modifications are highlighted in red.

Protocol 3 (A $(1,n)$-time program scheme)

For a security parameter κ, message space $\mathcal{M}$, input space $\mathcal{X}$, and token key space $\mathcal{K}$, let P_1 be the encapsulator, P_2 be the evaluator, $\mathcal{F}_{CT}$ be the ideal functionality of consumable tokens with negligible soundness error, $[\omega, d, \delta]_q$ be a linear code C with a generating matrix G such that $|C| = |\mathcal{X}|$, and $PRG : \{0,1\}^\kappa \to \{0,1\}^{\omega q |m|}$ be a psuedorandom generator, where $m \in \mathcal{M}$. Construct a tuple of algorithms $(\mathsf{Encap}, \mathsf{Eval})$ for a $(1,n)$-time program scheme as follows.

Encap: On input an arbitrary function f with input space $\mathcal{X}$, a linear code $[\omega, d, \delta]_q$ with generating matrix G, token key subspace K such that $|K| = q$, do the following:

1. Generate secret key $sk \in \{0,1\}^\kappa$ and a string $r \in \{0,1\}^\kappa$ both at random.
2. Generate messages $m_{i,j} = PRG(r)[i,j] \,\|\, \phi^{n(|x|+\ell_{out})}$ for all $i \in \{0, \ldots, \omega - 1\}$ and $j \in \{0, \ldots, q-1\}$.
3. For $i \in \{0, \ldots, \omega - 1\}$, generate a token ct_i, with a unique token ID tid_i, encoding messages $m_{i,0}, \ldots, m_{i,q-1}$ using $k_0 \ldots k_{q-1} \in K$. This is done by sending the command $(\mathsf{Encode}, \mathsf{tid}_i, \{k_0 \ldots k_{q-1}\}, \{m_{i,0}, \ldots, m_{i,q-1}\}, q)$ to $\mathcal{F}_{CT}$.
4. Obtain $eP = i\mathcal{O}(\mathsf{Prog}_{G,n,sk,r,f})$, where $\mathsf{Prog}_{G,n,sk,r,f}$ is defined in Figure 7.
5. Output $(\mathsf{ct}_0, \ldots, \mathsf{ct}_{\omega-1}, eP)$.

Eval: On input a $(1,n)$-$\mathsf{Prog} = (\mathsf{ct}_0, \ldots, \mathsf{ct}_{\omega-1}, eP)$, $x \in \mathcal{X}$, and a set of token keys $K = \{k_0 \ldots k_{q-1}\}$, do the following:

1. Map x to a codeword $\mathbf{c}$.
2. For each $i \in \{0, \ldots, \omega - 1\}$, query token ct_i using $k_{\mathbf{c}[i]}$ by sending the command $(\mathsf{Decode}, \mathsf{tid}_i, k_{\mathbf{c}[i]})$ to $\mathcal{F}_{CT}$ and get m_i in return.
3. Output $\mathsf{out} = eP(m_0 \,\|\, \cdots \,\|\, m_{\omega-1}, x)$.

Fig. 6: A modified construction for $(1,n)$-time programs.

Program $\mathsf{Prog}_{G,n,sk,r,f}$

Input: m, x
Description:

1. Parse m as $m_0 \,\|\, \cdots \,\|\, m_{\omega-1}$, and parse each m_i as $m_i^0 \,\|\, m_i^1$
2. Use G to compute the codeword $\mathbf{c}$ that corresponds to x.
3. Check that m corresponds to a valid codeword: Let $B = PRG(r)$, if $\exists B[i, \mathbf{c}[i]] \neq m_i^0$, then output $\perp$.
4. Set $y_i = \mathsf{Decrypt}(sk, m_i^1)$ for all $i \in \{0, \ldots, \omega - 1\}$.
5. If $\exists y_i \neq \phi^{n(|x|+\ell_{out})}$, then take the first such y_i and do the following:
   - Parse y_i as $y_i^0 \,\|\, \cdots \,\|\, y_i^{n-1}$.
   - Parse each y_i^j as $y_i^{j,0} \,\|\, y_i^{j,1}$ (for $j \in \{0, \ldots, n-1\}$).
   - Output $y_i^{j,1}$ for which $y_i^{j,0} = x$.

   Else, output $f(x)$.

Fig. 7: The program $\mathsf{Prog}_{G,n,sk,r,f}$ with linear error correcting codes.