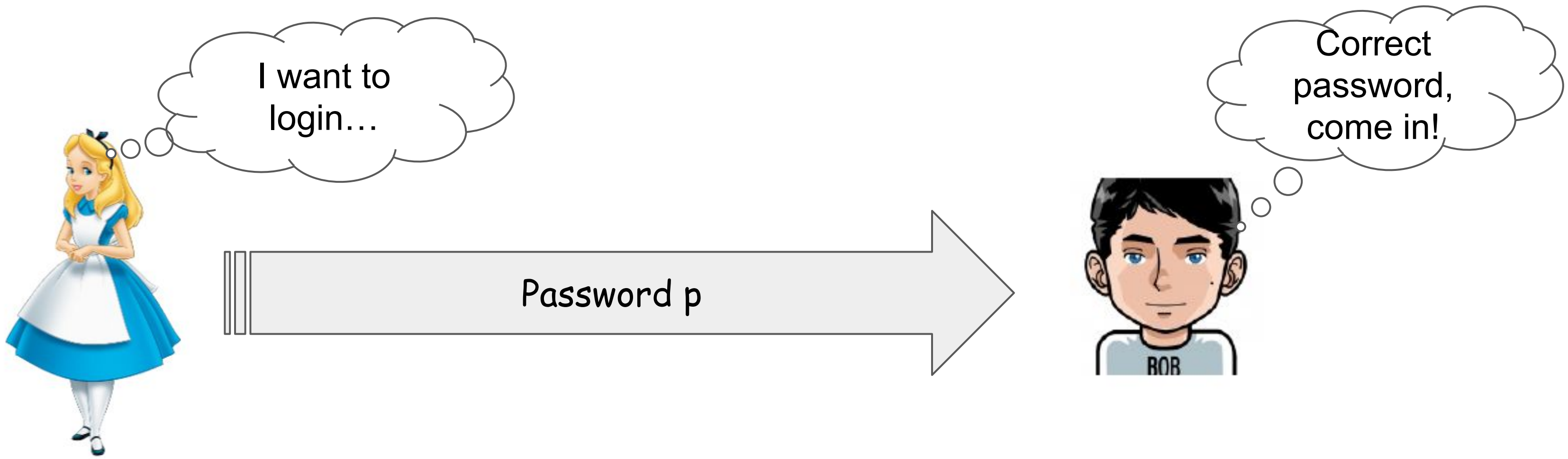
Password-authenticated Cryptography from Consumable Tokens

Ghada Almashaqbeh

University of Connecticut

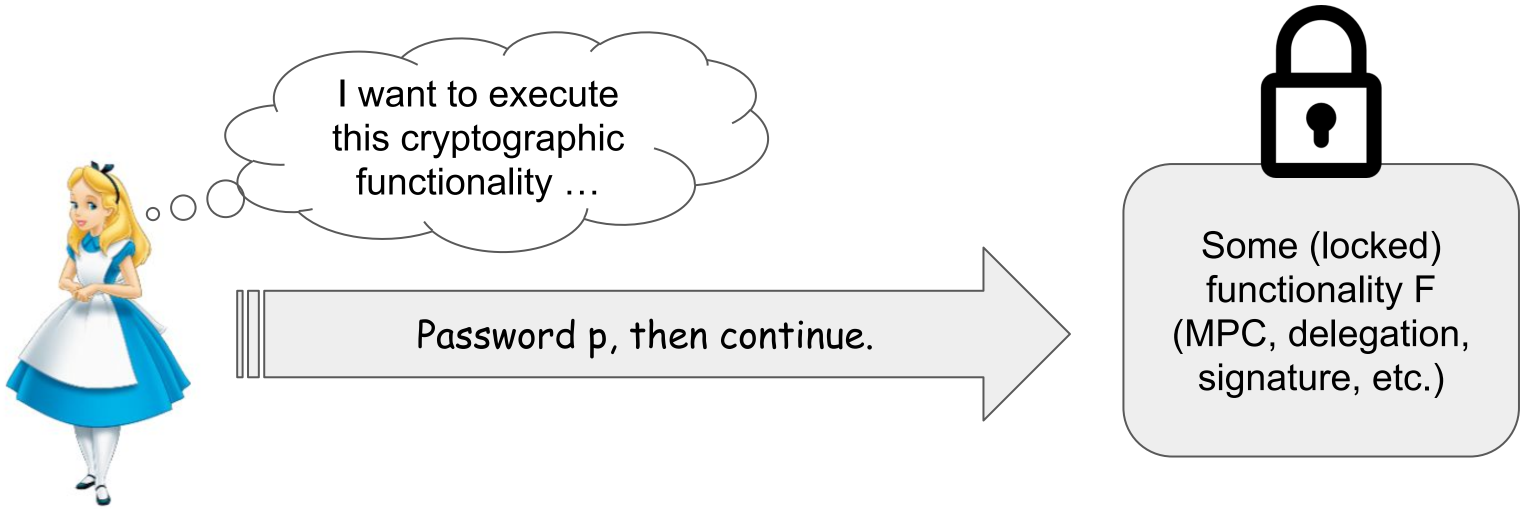
CSCML 2024

Password Authentication



- Passwords are widely adopted for user authentication in practice.
- Can we bootstrap a strongly-secure setting based on them?
  - This has been extensively studied for key exchange (PAKE).
  - Other instances include signatures, secret sharing, and encryption

Password-authenticated Cryptography



A unified notion in which knowing a password allows executing cryptographic functionalities.

- Resistant to exhaustive search over the password space.
  - without strong trust assumptions (such as interacting with a trusted entity or trusted hardware).

This work

New Models: PAD and PAMPC

- **Password-authenticated delegation (PAD):**
  - A party delegates her cryptographic power to another such that knowing a password permits exercising the delegation.
- **Password-authenticated multiparty computation (PAMPC):**
  - Participation, and hence, the MPC protocol execution, requires knowing a password.
- In both cases, an outsider can make a few password guesses.

Motivation

Consumable Tokens

- Self-destructed and unclonable memory gadgets.
  - Offers limited number of data retrievals.
    - Each retrieval consumes part of the token.
  - After *n* retrievals, the whole token is destructed.
- Recently, they have been instantiated using unclonable polymers, in particular, proteins.*

Can we utilize consumable tokens to realize PAD and PAMPC?

*G. Almashaqbeh, R. Canetti, Y. Erlich, J. Gershoni. T. Malkin, I. Pe'er, A. Roitburd-Berman, and E. Tromer, *Unclonable Polymers and Their Cryptographic Applications*, in Eurocrypt 2022.

Contributions

A formalization of the PAD and PAMPC models.

Consumable token-based constructions.

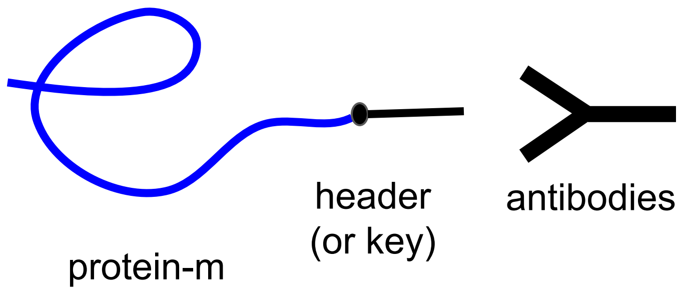
Open questions and future work directions.

Detour: Consumable Memory Tokens

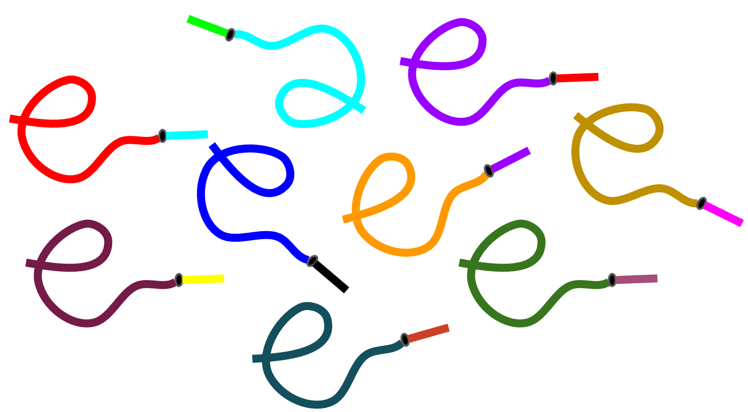
- Storing digital data in the form of protein material.
  - Inspired by DNA synthesis.
- Proteins provide additional features:
  - Proteins are unclonable; given a protein sample we cannot replicate it or get the genetic information out of it.
  - [Reading] proteins is destructive; sequencing a protein to retrieve the digital message, is destructive.
- The construction relies on these features to build consumable memory tokens.

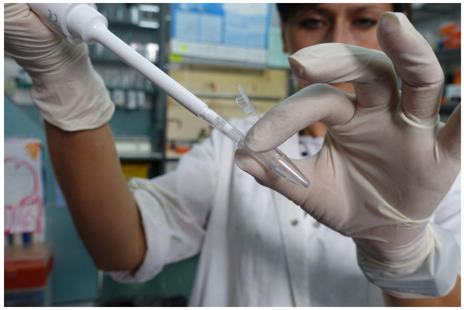
Data Storage

Synthesize m



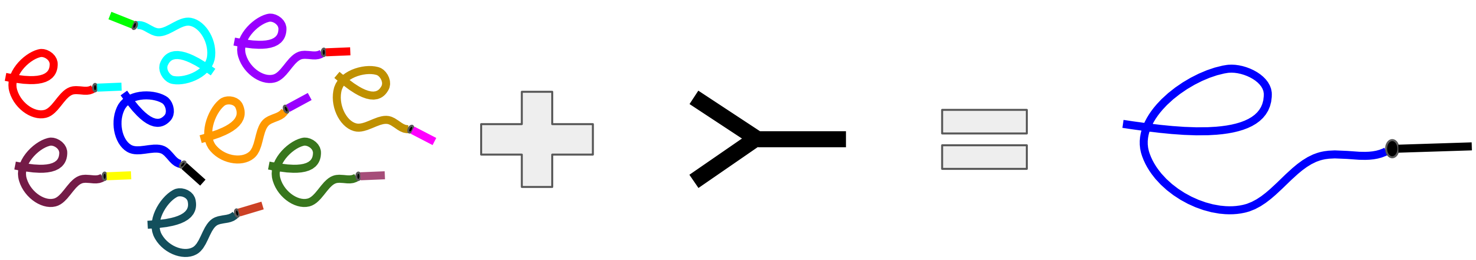
Mix with decoy proteins





Data Retrieval

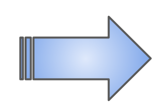
To retrieve m, first purify



then read the sequence

SYRGAA …

Amino acids



m

The Model

- Extension: Partial retrievable memory.
  - Storing multiple messages such that only a subset of them can be retrieved but not all of them.
- Limitations:
  - Non-negligible soundness error.
  - Power gap between the honest party and the adversary.
    - For each one honest retrieval query, the adversary can perform n queries.

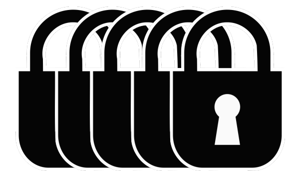
Applications

Bounded-query Digital lockers

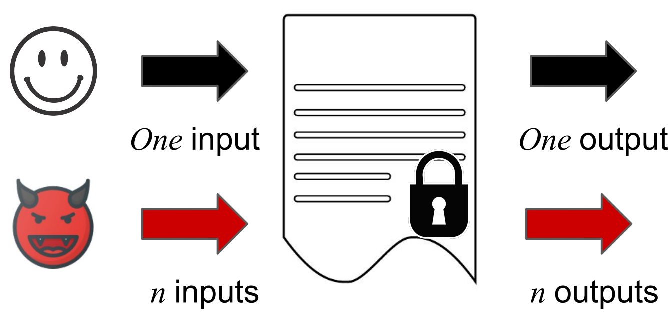
Password $p \in \mathcal{P}$ and message m

$c = Enc_p(m)$

$i \in \{1, \ldots, n\} : p_i \in \mathcal{P}, Dec_{p_i}(c)$



(1,n)-time programs

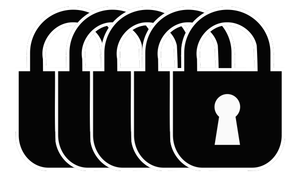


Applications

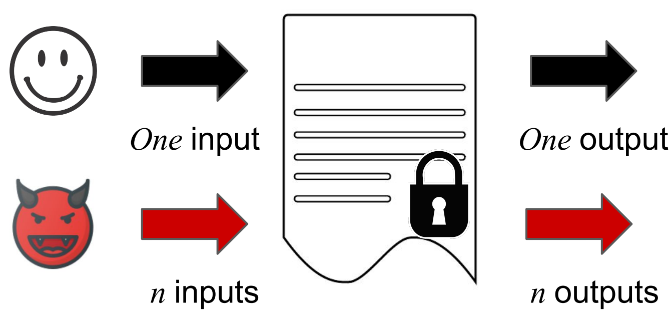
Bounded-query Digital lockers

Password $p \in \mathcal{P}$ and message m
$c = Enc_p(m)$

$i \in \{1, \ldots, n\} : p_i \in \mathcal{P}, Dec_{p_i}(c)$



(1,n)-time programs



Contributions

A formalization of the PAD and PAMPC models.

Consumable token-based constructions.

Open questions and future work directions.

The PAD Model

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.

The PAD Model

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.

Constructions I

- **Generic construction** that realizes any cryptographic capability.
- Combines bounded-query digital lockers and (1,n)-time programs.
  - They set of keys used for the (1,n)-time program consumable tokens is generated using the output of a PRG.
  - The PRG seed *s* is stored in the digital locker, without the password the key set cannot be generated.
- Downside: requires $i\mathcal{O}$.

Constructions II

- **Customized constructions**.
  - Basic idea: Encrypt the delegation information and store the decryption key in a bounded-query digital locker.
  - PAD for Signatures:
    - (Tokenized) proxy signatures ⇒ send encrypted tokens ⇒ p is needed to retrieve the decryption key and access the tokens.
    - Another construction based on Chameleon hash functions.

The PAMPC Model

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.

The PAMPC Model

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 $f : \{\{0,1\}^*\}^w \rightarrow \{0,1\}^*$, 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.

Constructions I

- **Password-authenticated two-party non-interactive MPC.**
  - A *password-authenticated non-interactive oblivious transfer* + Garbled circuits.
    - We formalize a model for this new OT notion and show a construction using consumable tokens.
  - P2 needs the password to retrieve the input labels of her input.
- Secure against semi-honest adversaries; malicious adversaries are problematic due to the power gap.
  - Malicious insider (i.e., corrupt P2) ⇒ not secure
  - Malicious outsider (i.e., does not know p) ⇒ depends on when p is guessed.

Constructions II

- **Password-authenticated interactive MPC.**
  - Secret sharing-based MPC.
  - A party sends a share of her input in a bounded-query digital locker.
  - Knowing p is needed to retrieve the shares needed to perform the MPC protocol.
- Secure against malicious (insider and outsider) adversaries.

Conclusion and Future Work Directions

- This work.
  - New models of password-authenticated cryptography — delegation and MPC.
  - Examined the power of consumable tokens in realizing these notions.
  - The power gap in these tokens impacted construction security.
- Future work.
  - Combine unclonable polymers with other technologies, such as quantum computing, to close the gap.

Thank you!

Questions?

ghada@uconn.edu
https://ghadaalmashaqbeh.github.io/