From Stateful to Resettable Hardware Using Symmetric Assumptions

Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade & Tobias Nilges

Aarhus University, TNG Technology Consulting GmbH, Karlsruhe Institute of Technology
**General Setting**

$\mathcal{F}$ is arbitrary functionality, e.g. OT, Commitment....
Motivation

• UC-secure protocols impossible without setup-assumptions
• [Katz07] introduced tamper-proof hardware as a UC setup-assumption
• Stateful token: statistically UC-secure OT is possible [DKM11]
What about resettable tokens?

- Still powerful, but most **statistically** secure protocols impossible [GIMS10]
- Feasibility of NI-2PC for resettable functionalities shown by [DMMN13]
- Open question: relation between stateful and resettable token protocols wrt feasibility?
Our Results

• All protocols based on stateful tokens can be transformed to use resettable tokens

• General compiler for UC-secure protocols
  – Requires interaction
  – Requires computational assumptions (only OWF!) or additional setup
Basic Idea

• Shift state from token to Alice:
  – Alice authenticates inputs
  – Bob sends authenticated value to token

• **Problem**: Alice must not learn Bob’s inputs

• **Solution**:
  – Alice authenticates encoding of input
  – Bob provides authentication and decoding information
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Interaction with Bob

**Bob**
- Interaction with token

---

Underlying protocol $\Pi_{\mathcal{F}}^{sf}$

Up to poly(k) times
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Interaction with Bob

**Bob**
- Interaction with token

**Token**
- Up to poly(k) times
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob

**Bob**
- Interaction with token

**Token**
Up to $\text{poly}(k)$ times
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob

**Bob**
- Answers only authenticated queries

**Token**
- Resettable token
- Up to $\text{poly}(k)$ times
- Interaction with token
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob
- Authentication step

**Bob**
- Interaction with Bob
- Authentication step

**Token**

- $\text{enc}(\text{inp})$
- $\sigma_{\text{inp}}$

Up to poly(k) times
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob
- Authentication step

**Bob**

**Token**
- Interaction with token

**Alice obliviously authenticates input**

- $\text{enc}(\text{inp})$
- $\sigma_{\text{inp}}$

Up to $\text{poly}(k)$ times
Blueprint of Our Approach

**Alice**
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob
- Authentication step

**Bob**
- Verify authentication
- Interaction with token

---

Token

Up to poly(k) times
Blueprint of Our Approach

*Alice*
- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob
- Authentication step

*Bob*
- Verify authentication
- Interaction with token

**New protocol $\Pi_{\mathcal{F}}^{res}$**

**Token**

Up to poly(k) times
Blueprint of Our Approach

**Alice**

- Generate token program $T$
- Enhanced program $T'$
- Interaction with Bob
- Authentication step

**Bob**

- Verify authentication
- Interaction with token

New protocol $\Pi^{res}_{\mathcal{F}}$

Token

Caution: channel from Alice to token

Up to poly(k) times
Two Solutions

• Using resettably-sound zero-knowledge
  – Non-black-box, but necessary [DMMN13, CKS+14]

• OT-hybrid model
  – Allows only a fixed number of messages
  – Inf.-th. transformation
Solution Based on resettably-sound ZK

**Alice**
- Generate $T$ according to $\Pi_{\mathcal{F}}^{sf}$
- Add 'functionality`

$\sigma \leftarrow \text{Sign}_{sgk}(c)$

**Bob**

$c \leftarrow \text{Commit}(inp; r)$

$\pi \leftarrow P(c, \sigma, r, inp)$

$r' \leftarrow \text{PRF}(inp)$

$V(\pi; r') = 1?$

$\text{out} \leftarrow T(inp)$

*Up to poly(k) times*
Proof Idea

• Every adversary against $\Pi^\text{res}_F$ can be transformed into adversary against $\Pi^\text{sf}_F$

• $\Pi^\text{sf}_F$ is UC-secure by assumption

• Corrupted Receiver:
  - Simulator has joint view of sender and token
  - Locally performs all checks that the token would perform
  - If checks are OK, proceed like in $\Pi^\text{sf}_F$

• Corrupted Sender:
  - Simulator has to input token code of $\Pi^\text{res}_F$ into stateful token
  - Simulator first constructs $\tilde{T}$:
    • Use source code of $T^{res}$ to create $V^*$
    • Use non-black-box simulator on $V^*$ to generate fake proof
    • Upon input, fake proof and proceed with execution of $T^{res}$
Efficiency

• ZK proof for each token input, but
  – Typically constant round protocols...
  – Some protocols allow non-adaptive inputs

• Non-adaptive inputs: create hash-tree of queries and authenticate root
  – CRHF > OWF!
  – Use Sig-Com Trees [CPS13], based on OWF
Implications

• Apply compiler to [DKM11] to obtain most efficient UC-secure OT-protocol from OWF
  – Token sent in one direction only
  – Constant round
  – Very efficient ([DKM11] provides inf-th. security)

• In OT-hybrid model, even inf-th. protocols can be realized
Thank You!