MPC for Group Reconstruction Circuits

Lúcás Críostóir Meier

June 8, 2022

Abstract

In this paper, we present a thing.

1 Introduction

Write the introduction

2 Background

Throughout this paper, we let $\mathbb{G}$ denote a group of prime order q, with generators G and H. Let $\mathbb{F}_q$ denote the scalar field associated with this group, and let $\mathbb{Z}/(q)$ denote the additive group of elements in this field.

We make heavy use of group homomorphisms throughout this paper. We let

$$\varphi(P_1, \ldots, P_m) : \mathbb{A} \to \mathbb{B}$$

denote a homomorphism from $\mathbb{A}$ to $\mathbb{B}$, parameterized by some public values $P_1, \ldots, P_m$. Commonly $\mathbb{A}$ will be a product of several groups $\mathbb{G}_1, \ldots, \mathbb{G}_n$, in which case we'd write:

$$\varphi(P_1, \ldots, P_m)(x_1, \ldots, x_n)$$

to denote the application of φ to an element $(x_1, \ldots, x_n)$ of the product group. Such an element is sometimes treated as a vector $\mathbf{x}$, in which case we write $\varphi(P_1, \ldots, P_m)(\mathbf{x})$. We also often leave the public values P_i implicit.

2.1 Pedersen Commitments

2.2 Sigma Protocols

2.3 Maurer's φ-Proof

In [Mau09], Maurer generalized Schnorr's sigma protocol for knowledge of the discrete logarithm [Sch90] to a much larger class of relations. In particular, Maurer provided a sigma protocol for proving knowledge of the pre-image of a group homomorphism φ. We denote this protocol as a "φ-proof", and recapitulate the scheme here.

Given a homomorphism $\varphi : \mathbb{A} \to \mathbb{B}$, and a public value $X \in \mathbb{B}$, the prover wants to demonstrate knowledge of a private value $x \in \mathbb{A}$ such that $\varphi(x) = X$. The prover does this by means of Protocol 2.1:

Protocol 2.1: φ-Proof

Prover		**Verifier**
knows $x \in \mathbb{A}$		public $X \in \mathbb{B}$
$k \xleftarrow{R} \mathbb{A}$		
$K \leftarrow \varphi(k)$		
	$\xrightarrow{K}$	
		$c \xleftarrow{R} \mathbb{Z}/(p)$
	$\xleftarrow{c}$	
$s \leftarrow k + c \cdot x$		
	$\xrightarrow{s}$	
		$\varphi(s) \stackrel{?}{=} K + c \cdot X$

Here, p is chosen such that $\forall B \in \mathbb{B}.\ p \cdot B = 0$. In practice, we'll set $p = q$, which will work perfectly for the groups we use, which are all products of $\mathbb{G}$ or $\mathbb{Z}/(q)$.

Claim 2.1. Protocol 2.1 is a valid sigma protocol.

Completeness follows directly from the fact that φ is a homomorphism.

For the HVZK property, the simulator $\mathcal{S}(X, c)$ works by generating a random $s \xleftarrow{R} \mathbb{A}$, and then setting $K := \varphi(S) - c \cdot X$.

Finally, we prove 2-extractability. Given two verifying transcripts (K, c, s) and (K, c', s') sharing the first message, we extract a value $\hat{x}$ satisfying $\varphi(\hat{x}) = X$ as follows:

$$
\begin{aligned}
\varphi(s) - c \cdot X = K &= \varphi(s') - c' \cdot X \\
\varphi(s) - \varphi(s') &= c \cdot X - c' \cdot X \\
\frac{1}{c - c'} \cdot \varphi(s - s') &= X \\
\varphi\left(\frac{s - s'}{c - c'}\right) &= X
\end{aligned}
$$

Thus, defining $\hat{x} := (s - s')/(c - c')$, we successfully extract a valid pre-image.

We conclude that the protocol is a valid sigma protocol.

■

Maurer's protocol can also work even in the case where the order of the groups are not known, but this makes the challenge generation a bit more complicated, and we don't need this functionality in this work.

2.4 UC Security and the Hybrid Model

2.5 Ideal Functionalities for Sigma Protocols

Functionality 2.1: Zero-Knowledge Functionality $\mathcal{F}(\texttt{ZK}, \varphi)$

A functionality $\mathcal{F}$ for parties $P_1, \ldots, P_n$.

On input $(\texttt{prove}, \texttt{sid}, X, x)$ from P_i:
$\mathcal{F}$ checks that $\texttt{sid}$ has not been used by P_i before.
$\mathcal{F}$ checks that $\varphi(x) \stackrel{?}{=} X$.
$\mathcal{F}$ generates a new token π, and sets $X_\pi \leftarrow X$.
$\mathcal{F}$ replies with $(\texttt{proof}, \pi)$.

On input $(\texttt{verify}, X, \pi)$:
$\mathcal{F}$ replies with $(\texttt{verify-result}, X_\pi \stackrel{?}{=} X)$.

2.6 Broadcast Functionalities

Functionality 2.2: Authenticated Broadcast Functionality $\mathcal{C}$

A functionality $\mathcal{C}$ for parties $P_1, \ldots, P_n$.

On receiving $(\texttt{broadcast-in}, \texttt{sid}, m)$ from P_i:
$\mathcal{C}$ checks that $\texttt{sid}$ has not been used by P_i before.
$\mathcal{C}$ sends $(\texttt{broadcast-out}, \texttt{pid}_i, \texttt{sid}, m)$ to every party P_j.

3 Group Reconstruction Circuits

3.1 Formal Definition

3.2 Normalized Form

4 MPC Protocol for GRCs

4.1 Ideal Functionality

Functionality 4.1: GRC functionality $\mathcal{F}(\texttt{GRC}, \Phi, \mathbf{X}^j, \mathbf{Y}^j)$

A functionality $\mathcal{F}$ for parties $P_1, \ldots, P_n$.

After receiving $(\texttt{input}, \texttt{sid}, \mathbf{x}^j, \mathbf{y}^j, \boldsymbol{\alpha}^j, \mathbf{k}^j)$ from every party P_j:
$\mathcal{F}$ checks, for every $j \in [n]$, that:

$$\begin{aligned} \mathbf{X}^j &\stackrel{?}{=} \mathbf{x}^j \cdot G \\ \mathbf{Y}^j &\stackrel{?}{=} \mathbf{y}^j \cdot G + \boldsymbol{\alpha}^j \cdot H \end{aligned}$$

$\mathcal{F}$ computes, for each round $r \in [d]$:

$$\begin{aligned} \mathbf{V}_r^j &:= \varphi_r(\mathbf{V}_1, \ldots, \mathbf{V}_{r-1})(\mathbf{x}^j, \mathbf{y}^j, \mathbf{k}^j) \\ \mathbf{V}_r &:= \sum_j \mathbf{V}_r^j \end{aligned}$$

$\mathcal{F}$ sends $(\texttt{output}, \texttt{sid}, \mathbf{V}_1^1, \ldots, \mathbf{V}_d^n)$ to every party P_j.

4.2 Protocol

$$\psi_r(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \mathbf{k}, \boldsymbol{\beta}) := (\varphi_r(\mathbf{x}, \mathbf{y}, \mathbf{j}), \mathbf{x} \cdot G, \text{Commit}(\mathbf{y}, \boldsymbol{\alpha}), \text{Commit}(\mathbf{k}, \boldsymbol{\beta}))$$

Protocol 4.1: MPC protocol for $\Phi, \mathbf{X}^j, \mathbf{Y}^j$

Each party P_j has inputs $\mathbf{x}^j$ and $\mathbf{y}^j$, committed to by $\mathbf{X}^j$ and $\mathbf{Y}^j$. They also have decommitments $\boldsymbol{\alpha}^j$ for $\mathbf{Y}^j$. Each party P_j also has a vector $\mathbf{k}^j$, which honest parties will have generated randomly.

Round 0
Each party P_j generates a random vector $\boldsymbol{\beta}^j$, and creates a commitment to $\mathbf{k}^j$ with:

$$\mathbf{K}^j := \mathrm{Commit}(\mathbf{k}^j, \boldsymbol{\beta}^j)$$

P_j sends $(\texttt{broadcast-in}, \texttt{sid}, \mathbf{K}^j)$ to the broadcast functionality $\mathcal{C}$.

Round r
Each party P_j computes $\mathbf{V}_r^j := \varphi_r(\mathbf{V}_1, \ldots, \mathbf{V}_{r-1})(\mathbf{x}^j, \mathbf{y}^j, \mathbf{k}^j)$.
Each party P_j sends $(\texttt{prove}, \texttt{sid}, (\mathbf{V}_r^j, \mathbf{X}^j, \mathbf{Y}^j, \mathbf{K}^j), (\mathbf{x}^j, \mathbf{y}^j, \boldsymbol{\alpha}^j, \mathbf{k}^j, \boldsymbol{\beta}^j))$ to $\mathcal{F}(\texttt{ZK}, \psi_r)$, receiving π_r^j in return.
Each party P_j sends $(\mathbf{V}_r^j, \pi_r^j)$ to every other party.

After receiving $(\mathbf{V}_r^i, \pi_r^i)$ from all other parties, P_j checks, for each i, that the proof is valid, by sending $(\texttt{verify}, (\mathbf{V}_r^i, \mathbf{X}^i, \mathbf{Y}^i, \mathbf{K}^i), \pi_r^i)$ to $\mathcal{F}(\texttt{ZK}, \psi_r)$, and aborting if the functionality returns 0.
Each party P_j then stores each $\mathbf{V}_r^i$ as part of its output.

4.3 Security Analysis

4.4 Practical Considerations

5 Applications

6 Limitations and Further Work

7 Conclusion

References

[Mau09] Ueli Maurer. Unifying Zero-Knowledge Proofs of Knowledge. In *AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 272–286. Springer, Berlin, Heidelberg, 2009.

[Sch90] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 239–252, New York, NY, 1990. Springer.