On the Complexity of Additively Homomorphic UC Commitments

Tore Kasper Frederiksen, Thomas P. Jakobsen,
Jesper Buus Nielsen, and Roberto Trifiletti(✉) [⋆⋆⋆]

Department of Computer Science, Aarhus University
{jot2re|tpj|jbn|roberto}@cs.au.dk

Abstract. We present a new constant round additively homomorphic commitment scheme with (amortized) computational and communication complexity linear in the size of the string committed to. Our scheme is based on the non-homomorphic commitment scheme of Cascudo *et al.* presented at PKC 2015. However, we manage to add the additive homomorphic property, while at the same time reducing the constants. In fact, when opening a large enough batch of commitments we achieve an amortized communication complexity converging to the length of the message committed to, i.e., we achieve rate 1 as the commitment protocol by Garay *et al.* from Eurocrypt 2014. A main technical improvement over the scheme mentioned above, and other schemes based on using error correcting codes for UC commitment, we develop a new technique which allows to based the extraction property on erasure decoding as opposed to error correction. This allows to use a code with significantly smaller minimal distance and allows to use codes without efficient decoding. Our scheme only relies on standard assumptions. Specifically we require a pseudorandom number generator, a linear error correcting code and an ideal oblivious transfer functionality. Based on this we prove our scheme secure in the Universal Composability (UC) framework against a static and malicious adversary corrupting any number of parties.

On a practical note, our scheme improves significantly on the non-homomorphic scheme of Cascudo *et al.* Based on their observations in regards to efficiency of using linear error correcting codes for commitments we conjecture that our commitment scheme might in practice be more efficient than all existing constructions of UC commitment, even non-homomorphic constructions and even constructions in the random oracle model. In particular, the amortized price of computing one of our commitments is less than that of evaluating a hash function once.

Keywords: Commitments, UC, Homomorphic, Minimal Assumptions, Linear Error Correcting Codes, Erasure Codes.

[⋆] The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation and from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council.

[⋆⋆] Partially supported by the European Research Commission Starting Grant 279447.

1 Introduction

Commitment schemes are the digital equivalent of a securely locked box: it allows a sender P_s to hide a secret from a receiver P_r by putting the secret inside the box, sealing it, and sending the box to P_r. As the receiver cannot look inside we say that the commitment is *hiding*. As the sender is unable to change his mind as he has given the box away we say the commitment is also *binding*. These simple, yet powerful properties are needed in countless cryptographic protocols, especially when guaranteeing security against a *malicious* adversary who can arbitrarily deviate from the protocol at hand. In the stand-alone model, commitment schemes can be made very efficient, both in terms of communication and computation and can be based entirely on the existence of one-way functions. These can *e.g.* be constructed from cheap symmetric cryptography such as pseudorandom generators [Nao90].

In this work we give a commitment scheme secure in the UC-framework of [Can01], a model considering protocols running in a concurrent and asynchronous setting. The first UC-secure commitment schemes were given in [CF01, CLOS02] as feasibility results, while in [CF01] it was also shown that UC-commitments cannot be instantiated in the standard model and therefore require some form of setup assumption, such as a CRS. Moreover a construction for UC-commitments in such a model implies public-key cryptography [DG03]. Also, in the UC setting the previously mentioned hiding and binding properties are augmented with the notions of *equivocality* and *extractability*, respectively. (These properties are needed to realize the commitment functionality we introduce later on.) Loosely speaking, a scheme is equivocal if a single commitment can be opened to any message using special trapdoor information. Likewise a scheme is extractable if from a commitment the underlying message can be extracted efficiently using again some special trapdoor information.

Based on the above it is not surprising that UC-commitments are significantly less efficient than constructions in the stand-alone model. Nevertheless a plethora of improvements have been proposed in the literature, *e.g.* [DN02, NFT09, Lin11, BCPV13, Fuj14] considering different number theoretic hardness assumptions, types of setup assumption and adversarial models. Until recently, the most efficient schemes for the adversarial model we consider in this work was that of [Lin11, BCPV13] which is based on the DDH assumption in the CRS model.

Related Work. In [GIKW14] and independently in [DDGN14] it was considered to construct UC-commitments in the OT-hybrid model and at the same time confining the use of the OT primitive to a once-and-for-all setup phase. After the setup phase, the idea is to only use cheap symmetric primitives for each commitment thus amortizing away the cost of the initial OTs. Both approaches strongly resembles the "MPC-in-the-head" line of work of [IKOS07, HIKN08, IPS08] in that the receiver is watching a number of communication channels not disclosed to the sender. In order to cheat meaningfully in this paradigm the sender needs to cheat in many channels, but since he is unaware where the receiver is watching he will get caught with high probability. Concretely these schemes build on VSS and allow the receiver to learn an unqualified set of shares for a secret s. However the setup is such that the sender does not know which unqualified set is being "watched", so when opening he is forced to open to enough positions with consistent shares to avoid getting caught. The scheme of [GIKW14] focused primarily on the rate of the commitments in an asymptotic setting while [DDGN14] focused on the computational complexity. Furthermore the secret sharing scheme of the latter is based on Reed-Solomon codes and the scheme achieved both additive and multiplicative homomorphisms.

The idea of using OTs and error correction codes to realize commitments was also considered in [FJN$^+$13] in the setting of two-party secure computation using garbled circuits. Their scheme also allowed for additively homomorphic operations on commitments, but requires a code with a specific privacy property. The authors pointed to [CC06] for an example of such a code, but it turns out this achieves quite low constant rate due to the privacy restriction. Care also has to be taken when using this scheme, as binding is not guaranteed for all committed messages. The authors capture this by allowing some message to be "wildcards". However, in their application this is acceptable and properly dealt with.

Finally in [CDD$^+$15] a new approach to the above OT watch channel paradigm was proposed. Instead of basing the underlying secret sharing scheme on a threshold scheme the authors proposed a scheme for a particular access structure. This allowed realization of the scheme using additive secret sharing and any linear code, which achieved very good concrete efficiency. The only requirement of the code is that it is linear and the minimum distance is at least $2s+1$ for statistical security s. To commit to a message m it is first encoded into a codeword c. Then each field element c_i of c is additively shared into two field elements c_i^0 and c_i^1 and the receiver learns one of these shares via an oblivious transfer. This in done in the watch-list paradigm where the same shares c_i^0 are learned for all the commitments, by using the OTs only to transfer short seeds and then generating the share c_i^0 and c_i^1 for all commitments from these pairs of seeds. This can be seen as reusing an idea ultimately going back to [CvT95] in the context of the watch list paradigm. Even if the adversary commits to a string c' which is not a codeword, to open to another message, it would have to guess at least s of the random choice bits of the receiver. Furthermore the authors propose an additively homomorphic version of their scheme, however at the cost of resorting to using a VSS scheme along with imposing higher constants than their basic non-homomorphic construction.

Our Contribution. We introduce a new, very efficient, additively homomorphic UC-secure commitment scheme in the $\mathcal{F}_{\mathsf{ROT}}$-hybrid model which is described in Fig. 1. Our scheme shows that:

1. The asymptotic complexity of additively homomorphic UC commitment is the same as the asymptotic complexity of non-homomorphic UC commitment, i.e., the achievable rate is $1-o(1)$. In particular, the homomorphic property comes for free.
2. In addition to being asymptotically optimal, our scheme is also more practical (smaller hidden constants) than any other existing UC commitment scheme, even non-homomorphic schemes and even schemes in the random oracle model.

In more detail our main contributions are as follows:

– We improve on the basic non-homomorphic commitment scheme of [CDD$^+$15] by reducing the requirement of the minimum distance of the underlying linear code from $2s+1$ to s for statistical security s. At the same time our scheme becomes additively homomorphic, a property not shared with the above scheme. This is achieved by introducing an efficient consistency check at the end of the commit phase, as described now. Assume that the corrupted sender commits to a string c' which has Hamming distance 1 to some codeword c_0 encoding message m_0 and has Hamming distance $s-1$ to some other codeword c_1 encoding message m_1. For both the scheme in [CDD$^+$15] and our scheme this means the adversary can later open to m_0 with probability $\frac{1}{2}$ and to m_1 with probability 2^{-s+1}. Both of these probabilities are considered too high as we want statistical security 2^{-s}. So, even if we could decode c' to for instance m_0, this might not be the message that the adversary will open to later. It is, however, the case that the adversary cannot later open to both m_0 and m_1, except with probability 2^{-s} as this would require guessing s of the random choice bits. The UC simulator, however, needs to extract which of m_0 and m_1 will be opened to already at commitment time. We introduce a new consistency check where we after the commitment phase ask the adversary to open a random linear combination of the committed purported codewords. This linear combination will with overwhelming probability in a well defined manner "contain" information about every dirty codeword c' and will force the adversary to guess some of the choice bits to successfully open it to some close codeword c. The trick is then that the simulator can extract which of the choice bits the adversary had to guess and that if we puncture the code and the committed strings at the positions at which the adversary guessed the choice bits, then the remaining strings can be proven to be codewords in the puncture code. Since the adversary guesses at most $s-1$ choice bits, except with negligible probability 2^{-s} we only need to puncture $s-1$ positions, so the punctured code still has distance 1. We can therefore erasure decode and thus extract the committed message. If the adversary later open to another message he will have to guess additional choice bits, bringing him up to having guessed at least s choice bits. With the minimal distance lowered the required code length is also reduced and therefore also the amount of required initial OTs. As an example, for committing to messages of size $k=256$ with statistical security $s=40$ this amounts to roughly 33% less initial OTs than required by [CDD+15].

- We furthermore propose a number of optimizations that reduce the communication complexity by a factor of 2 for each commitment compared to [CDD+15] (without taking into account the smaller code length required). We give a detailed comparison to the schemes of [Lin11, BCPV13] and [CDD+15] in Section 4 and show that for the above setting with $k=256$ and $s=40$ our new construction outperforms all existing schemes if committing to 304 messages or more while retaining the computational efficiency of [CDD+15]. This comparison includes the cost of the initial OTs. If committing to 10,000 messages or more we see the total communication is around 1/3 of [BCPV13], around 1/2 of the basic scheme of [CDD+15] and less than 1/21 of the homomorphic version.
- Finally we give an extension of any additively homomorphic commitment scheme that achieves an amortized rate close to 1 in the opening phase. Put together with our proposed scheme and breaking a long message into many smaller blocks we achieve close to rate 1 in both the commitment and open phase of our protocol. This extension is interactive and is very similar in nature to the introduced consistency check for decreasing the required minimum distance. Although based on folklore techniques this extension allows for very efficiently homomorphic commitment to long messages without requiring correspondingly many OTs.

2 The Protocol

We use κ and s to denote the computational and statistical security parameter respectively. This means that for any fixed s and any polynomial time bounded adversary, the advantage of the adversary is $2^{-s}+\mathrm{negl}(\kappa)$ for a negligible function negl. I.e., the advantage of any adversary goes to 2^{-s} faster than any inverse polynomial in the computational security parameter. If $s=\Omega(\kappa)$ then the advantage is negligible. We will be working over an arbitrary finite field $\mathbb{F}$. Based on this, along with s, we define $\hat{s}=\lceil s/\log_2(|\mathbb{F}|)\rceil$.

$\mathcal{F}_{\mathsf{ROT}}$ interacts with a sender P_s, a receiver P_r and an adversary $\mathcal{S}$ and it proceeds as follows:

Transfer: Upon receiving $(\texttt{transfer},\texttt{sid},\texttt{otid},k)$ from both P_s and P_r, forward this message to $\mathcal{S}$ and wait for a reply. If $\mathcal{S}$ sends back $(\texttt{no-corrupt},\texttt{sid},\texttt{otid})$, sample $l^0,l^1 \in_R \{0,1\}^k$ and $b \in_R \{0,1\}$ and output $\big(\texttt{deliver},\texttt{sid},\texttt{otid},\big(l^0,l^1\big)\big)$ to P_s and $\big(\texttt{deliver},\texttt{sid},\texttt{otid},\big(l^b,b\big)\big)$ to P_r.

If $\mathcal{S}$ instead sends back $\big(\texttt{corrupt-sender},\texttt{sid},\texttt{otid},\big(\tilde{l}^0,\tilde{l}^1\big)\big)$ or $\Big(\texttt{corrupt-receiver},\texttt{sid},\texttt{otid},\Big(\tilde{l}^{\tilde{b}},\tilde{b}\Big)\Big)$ and the sender, respectively the receiver is corrupted, proceed as above, but instead of sampling all values at random, use the values provided by $\mathcal{S}$.

Fig. 1. Ideal Functionality $\mathcal{F}_{\mathsf{ROT}}$.

We will use as shorthand $[n]=\{1,2,\ldots,n\}$, and $e\in_R S$ to mean: sample an element e uniformly at random from the set S. When $\boldsymbol{r}$ and $\boldsymbol{m}$ are vectors we write $\boldsymbol{r}\|\boldsymbol{m}$ to mean the vector that is the concatenation of $\boldsymbol{r}$ and $\boldsymbol{m}$. We write $y\leftarrow P(x)$ to mean: perform the (potentially randomized) procedure P on input x and store the output in variable y. We will use $x:=y$ to denote an assignment of x to y. We will interchangeably use subscript and bracket notation to denote an index of a vector, i.e. x_i and $\boldsymbol{x}[i]$ denotes the i'th entry of a vector $\boldsymbol{x}$ which we will always write in bold. Furthermore we will use $\pi_{i,j}$ to denote a projection of a vector that extracts the entries from index i to index j, *i.e.* $\pi_{i,j}(\boldsymbol{x})=(x_i,x_{i+1},\ldots,x_j)$. We will also use $\pi_l(\boldsymbol{x})=\pi_{1,l}(\boldsymbol{x})$ as shorthand to denote the first l entries of $\boldsymbol{x}$.

In Fig. 2 we present the ideal functionality $\mathcal{F}_{\mathsf{HCOM}}$ that we UC-realize in this work. The functionality differs from other commitment functionalities in the literature by only allowing the sender P_s to decide the number of values he wants to commit to. The functionality then commits him to *random* values towards a receiver P_r and reveals the values to P_s. The reason for having the functionality commit to several values at a time is to reflect the batched nature of our protocol. That the values committed to are random is a design choice to offer flexibility for possible applications. In Appendix A we show an efficient black-box extension of $\mathcal{F}_{\mathsf{HCOM}}$ to chosen-message commitments.

$\mathcal{F}_{\mathsf{HCOM}}$ interacts with a sender P_s, a receiver P_r and an adversary $\mathcal{S}$, working over a field $\mathbb{F}$.

Init: Upon receiving a message $(\texttt{init},\texttt{sid},k)$ from both parties P_s and P_r, store the message length k.

Commit: Upon receiving a message $(\texttt{commit},\texttt{sid},\eta)$ from P_s, forward this message to $\mathcal{S}$ and wait for a reply. If $\mathcal{S}$ sends back $(\texttt{no-corrupt},\texttt{sid})$ proceed as follows:
Sample η uniformly random values $\boldsymbol{r}_j \in \mathbb{F}^k$ and associate to each of these a unique unused identifier j and store the tuple $(\texttt{random},\texttt{sid},j,\boldsymbol{r}_j)$. We let $\mathcal{J}$ denote the set of these identifiers. Finally send $\left(\texttt{committed},\texttt{sid},\mathcal{J},\{\boldsymbol{r}_j\}_{j\in\mathcal{J}}\right)$ to P_s and $(\texttt{receipt},\texttt{sid},\mathcal{J})$ to P_r and $\mathcal{S}$.
If P_s is corrupted and $\mathcal{S}$ instead sends back $\left(\texttt{corrupt-commit},\texttt{sid},\{\tilde{\boldsymbol{r}}_j\}_{j\in\mathcal{J}}\right)$, proceed as above, but instead of sampling the values at random, use the values provided by $\mathcal{S}$.

Open: Upon receiving a message $\left(\texttt{open},\texttt{sid},\{(c,\alpha_c)\}_{c\in C}\right)$ from P_s, if for all $c \in C$, a tuple $(\texttt{random},\texttt{sid},c,\boldsymbol{r}_c)$ was previously recorded and $\alpha_c \in \mathbb{F}$, send $\left(\texttt{opened},\texttt{sid},\{(c,\alpha_c)\}_{c\in C},\sum_{c\in C}\alpha_c \cdot \boldsymbol{r}_c\right)$ to P_r and $\mathcal{S}$. Otherwise, ignore.

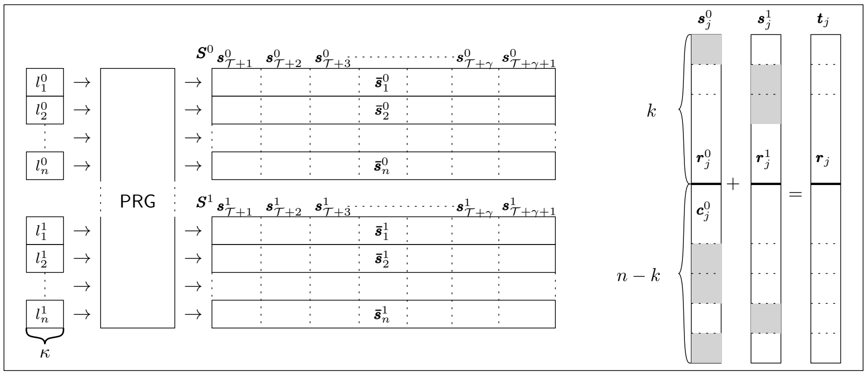
Fig. 2. Ideal Functionality $\mathcal{F}_{\mathsf{HCOM}}$.

2.1 Protocol Π_{HCOM}

Our protocol Π_{HCOM} is cast in the $\mathcal{F}_{\mathsf{ROT}}$-hybrid model, meaning the parties are assumed access to the ideal functionality $\mathcal{F}_{\mathsf{ROT}}$ in Fig. 1. The protocol UC-realizes the functionality $\mathcal{F}_{\mathsf{HCOM}}$ and is presented in full in Fig. 4 and Fig. 5. At the start of the protocol a once-and-for-all **Init** step is performed where P_s and P_r only need to know the size of the committed values k and the security parameters. We furthermore assume that the parties agree on a $[n,k,d]$ linear code $\mathcal{C}$ in systematic form over the finite field $\mathbb{F}$ and require that the minimum distance $d \geq s$ for statistical security parameter s. The parties then invoke n copies of the ideal functionality $\mathcal{F}_{\mathsf{ROT}}$ with the computational security parameter κ as input, such that P_s learns n pairs of κ-bit strings l_i^0,l_i^1 for $i \in [n]$, while P_r only learns one string of each pair. In addition to the above the parties also introduce a commitment counter $\mathcal{T}$ which simply stores the number of values committed to. Our protocol is phrased such that multiple commitment phases are possible after the initial ROTs have been performed, and the counter is simply incremented accordingly.

Next a **Commit** phase is described where at the end, P_s is committed to γ pseudorandom values. The protocol instructs the parties to expand the previously learned κ-bit strings, using a pseudorandom generator PRG, into row-vectors $\bar{\boldsymbol{s}}_i^b \in \mathbb{F}^{\mathcal{T}+\gamma+1}$ for $b \in \{0,1\}$ and $i \in [n]$. The reason for the extra length will be apparent later. We denote by $\mathcal{J} = \{\mathcal{T}+1,...,\mathcal{T}+\gamma+1\}$ the set of indices of the $\gamma+1$ commitments being setup in this invocation of **Commit**. After the expansion P_s knows all of the above $2n$ row-vectors, while P_r only knows half. The parties then view these row-vectors as matrices $\boldsymbol{S}^0$ and $\boldsymbol{S}^1$ where row i of $\boldsymbol{S}^b$ consists of the vector $\bar{\boldsymbol{s}}_i^b$. We let $\boldsymbol{s}_j^b \in \mathbb{F}^n$ denote the j'th column vector of the matrix $\boldsymbol{S}^b$ for $j \in \mathcal{J}$. These column vectors now determine the committed pseudorandom values, which we define as $\boldsymbol{r}_j = \boldsymbol{r}_j^0 + \boldsymbol{r}_j^1$ where $\boldsymbol{r}_j^b = \pi_k(\boldsymbol{s}_j^b)$ for $j \in \mathcal{J}$. The above steps are also pictorially described in Fig. 3.

The goal of the commit phase is for P_r to hold one out of two shares of each entry of a codeword of $\mathcal{C}$ that encodes the vector $\boldsymbol{r}_j$ for all $j \in \mathcal{J}$. At this point of the protocol, what P_r holds is however not of the above form. Though, because the code is in systematic form we have by definition that P_r holds such a sharing for the first k entries of each of these codewords. To ensure the same for the rest of the entries, for all $j \in \mathcal{J}$, P_s computes $\boldsymbol{t}_j \leftarrow \mathcal{C}(\boldsymbol{r}_j)$ and lets $\boldsymbol{c}_j^0 = \pi_{k+1,n}(\boldsymbol{s}_j^0)$. It then computes the correction value $\bar{\boldsymbol{c}}_j = \pi_{k+1,n}(\boldsymbol{t}_j) - \boldsymbol{c}_j^0 - \pi_{k+1,n}(\boldsymbol{s}_j^1)$ and sends this to P_r.



l_1^0
l_2^0
l_n^0
l_1^1
l_2^1
l_n^1
κ
PRG
$\boldsymbol{S}^0$ $\boldsymbol{s}^0_{\mathcal{T}+1}$ $\boldsymbol{s}^0_{\mathcal{T}+2}$ $\boldsymbol{s}^0_{\mathcal{T}+3}$ $\boldsymbol{s}^0_{\mathcal{T}+\gamma}$ $\boldsymbol{s}^0_{\mathcal{T}+\gamma+1}$
$\bar{\boldsymbol{s}}_1^0$
$\bar{\boldsymbol{s}}_2^0$
$\bar{\boldsymbol{s}}_n^0$
$\boldsymbol{S}^1$ $\boldsymbol{s}^1_{\mathcal{T}+1}$ $\boldsymbol{s}^1_{\mathcal{T}+2}$ $\boldsymbol{s}^1_{\mathcal{T}+3}$ $\boldsymbol{s}^1_{\mathcal{T}+\gamma}$ $\boldsymbol{s}^1_{\mathcal{T}+\gamma+1}$
$\bar{\boldsymbol{s}}_1^1$
$\bar{\boldsymbol{s}}_2^1$
$\bar{\boldsymbol{s}}_n^1$
$\boldsymbol{s}_j^0$
$\boldsymbol{s}_j^1$
$\boldsymbol{t}_j$
k
$n-k$
$\boldsymbol{r}_j^0$
$\boldsymbol{c}_j^0$
$\boldsymbol{r}_j^1$
$\boldsymbol{r}_j$
+
=

Fig. 3. On the left hand side we see how the initial part of the generate phase of Π_{HCOM} is performed by P_s when committing to γ messages. On the right hand side we look at a single column of the two matrices $\boldsymbol{S}^0, \boldsymbol{S}^1$ and how they define the resulting codeword $\boldsymbol{t}_j$ for column $j \in \mathcal{J}$, where $\mathcal{J} = \{\mathcal{T}+1, ..., \mathcal{T}+\gamma+1\}$.

When receiving the correction value $\bar{\boldsymbol{c}}_j$, we notice that for the columns $\boldsymbol{s}_j^0$ and $\boldsymbol{s}_j^1$, P_r knows only the entries $w_j^i = \boldsymbol{s}_j^{b_i}[i]$ where b_i is the choice-bit it received from $\mathcal{F}_{\mathsf{ROT}}$ in the i'th invocation. For all $l \in [n-k]$, if $b_{k+l} = 1$ it is instructed to update its entry as follows:

$$w_j^{k+l} := \bar{\boldsymbol{c}}_j[l] + w_j^{k+l} = \boldsymbol{t}_j[k+l] - \boldsymbol{c}_j^0[l] - \boldsymbol{s}_j^1[k+l] + w_j^{k+l} = \boldsymbol{t}_j[k+l] - \boldsymbol{c}_j^0[l] \ .$$

Due to the above corrections, it is now the case that for all $l \in [n-k]$ if $b_{k+l} = 0$, then $w_j^{k+l} = \boldsymbol{c}_j^0[l]$ and if $b_{k+l} = 1$, $w_j^{k+l} = \boldsymbol{t}_j[k+l] - \boldsymbol{c}_j^0[l]$. This means that at this point, for all $j \in \mathcal{J}$ and all $i \in [n]$, P_r holds exactly one out of two shares for each entry of the codeword $\boldsymbol{t}_j$ that encodes the vector $\boldsymbol{r}_j$.

The **Open** procedure is described in Fig. 5. It describes how P_s can open to linear combinations of previously committed values. We let C be the indices to be opened to and α_c for $c \in C$ be the corresponding coefficients. The sender then computes $\boldsymbol{r}^0 = \sum_{c \in C} \alpha_c \cdot \boldsymbol{r}_c^0$, $\boldsymbol{r}^1 = \sum_{c \in C} \alpha_c \cdot \boldsymbol{r}_c^1$, and $\boldsymbol{c}^0 = \sum_{c \in C} \alpha_c \cdot \boldsymbol{c}_c^0$ and sends these to P_r. When receiving the three values, the receiver computes the codeword $\boldsymbol{t} \leftarrow \mathcal{C}(\boldsymbol{r}^0 + \boldsymbol{r}^1)$ and from $\boldsymbol{c}^0$ and $\boldsymbol{t}$ it computes $\boldsymbol{c}^1$. It also computes $\boldsymbol{w} = \sum_{c \in C} \alpha_c \cdot \boldsymbol{w}_c$ and verifies that $\boldsymbol{r}^0, \boldsymbol{r}^1, \boldsymbol{c}^0$, and $\boldsymbol{c}^1$ are consistent with these. If everything matches it accepts $\boldsymbol{r}^0 + \boldsymbol{r}^1$ as the value opened to.

If the sender P_s behaves honestly in **Commit** of Π_{HCOM}, then the scheme is UC-secure as it is presented until now. In fact it is also additively homomorphic due to the linearity of the code $\mathcal{C}$ and the linearity of additive secret sharing. However, this only holds because P_r holds shares of valid codewords. If we consider a malicious corruption of P_s, then the shares held by P_r might not be of a valid codeword, and then it is undefined at commitment time what the value committed to is.

2.2 Optimizations over [CDD$^+$15]

The work of [CDD$^+$15] describes two commitment schemes, a basic and a homomorphic version. For both schemes therein the above issue of sending correct shares is handled by requiring the underlying code $\overline{\mathcal{C}}$ with parameters $[\overline{n}, k, \overline{d}]$ to have minimum distance $\overline{d} \geq 2s+1$, as then the committed values are always defined to be the closest valid codewords of the receivers shares. This is however not enough to guarantee binding when allowing homomorphic operations. To support this, the authors propose a version of the scheme that involves the sender P_s running a "MPC-in-the-head" protocol based on a verifiable secret sharing scheme of which the views of the simulated parties must be sent to P_r.

Up until now the scheme we have described is very similar to the basic scheme of [CDD$^+$15]. The main difference is the use of $\mathcal{F}_{\mathsf{ROT}}$ as a starting assumption instead of $\mathcal{F}_{\mathsf{OT}}$ and the way we define and send the committed value corrections. In [CDD$^+$15] the corrections sent are for both the 0 and the 1 share. This means they send $2\overline{n}$ field elements for each commitment in total. Having the code in systematic form implies that for all $j \in \mathcal{J}$ and $i \in [k]$ the entries $\boldsymbol{w}_j^i$ are already defined for P_r as part of the output of the PRG, thus saving $2k$ field elements of communication per commitment. Together with only sending corrections to the 1-share, we only need to send $n-k$ field elements as corrections. Meanwhile this only commits the sender to a pseudorandom value, so to commit to a chosen value another correction of k elements need to be sent. In total we therefore save a factor 2 of communication from these optimizations.

Π_{HCOM} describes a protocol between a sender P_s and a receiver P_r. We let $\mathsf{PRG}:\{0,1\}^{\kappa}\rightarrow\mathbb{F}^{\mathrm{poly}(\kappa)}$ be a pseudorandom generator with arbitrary polynomial stretch taking a κ-bit seed as input and outputting elements of a predetermined finite field $\mathbb{F}$.

Init:

1. On common input $(\texttt{init},\texttt{sid},k)$ we assume the parties agree on a linear code $\mathcal{C}$ in systematic form over $\mathbb{F}$ with parameters $[n,k,d]$. The parties also initialize an internal commitment counter $\mathcal{T}=0$.
2. For $i\in[n]$, both parties send $(\texttt{transfer},\texttt{sid},i,\kappa)$ to the functionality $\mathcal{F}_{\mathsf{ROT}}$. It replies with $\left(\texttt{deliver},\texttt{sid},i,\left(l_i^0,l_i^1\right)\right)$ to P_s and $\left(\texttt{deliver},\texttt{sid},i,\left(l_i^{b_i},b_i\right)\right)$ to P_r.

Commit:

Construct

1. On common input $(\texttt{commit},\texttt{sid},\gamma)$, for $i\in[n]$, both parties use PRG to extend their received seeds into vectors of length $\mathcal{T}+\gamma+1$. We denote the extended strings P_s holds by $\bar{\boldsymbol{s}}_i^0,\bar{\boldsymbol{s}}_i^1\in\mathbb{F}^{\mathcal{T}+\gamma+1}$ and we denote the extended string P_r holds by $\bar{\boldsymbol{s}}_i^{b_i}\in\mathbb{F}^{\mathcal{T}+\gamma+1}$. Next define the matrices $\boldsymbol{S}^0,\boldsymbol{S}^1\in\mathbb{F}^{n\times(\mathcal{T}+\gamma+1)}$ such that for $i\in[n]$ the i'th row of $\boldsymbol{S}^b$ is $\bar{\boldsymbol{s}}_i^b$ for $b\in\{0,1\}$.
2. Let $\mathcal{J}=\{\mathcal{T}+1,...,\mathcal{T}+\gamma+1\}$. For $j\in\mathcal{J}$ let the column vector of these matrices be $\boldsymbol{s}_j^0$, respectively $\boldsymbol{s}_j^1$. For $b\in\{0,1\}$, P_s lets $\boldsymbol{r}_j^b=\pi_k\left(\boldsymbol{s}_j^b\right)$ and lets $\boldsymbol{r}_j=\boldsymbol{r}_j^0+\boldsymbol{r}_j^1$. Also P_r lets $\boldsymbol{w}_j=\left(w_j^1,...,w_j^n\right)$ and $(b_1,...,b_n)\leftarrow\boldsymbol{b}$ where $w_j^i=\boldsymbol{s}_j^{b_i}[i]$ for $i\in[n]$.
3. For $j\in\mathcal{J}$, P_s computes $\boldsymbol{t}_j\leftarrow\mathcal{C}\left(\boldsymbol{r}_j\right)$ and lets $\boldsymbol{c}_j^0=\pi_{k+1,n}\left(\boldsymbol{s}_j^0\right)$. It then computes the correction value $\bar{\boldsymbol{c}}_j=\pi_{k+1,n}\left(\boldsymbol{t}_j\right)-\boldsymbol{c}_j^0-\pi_{k+1,n}\left(\boldsymbol{s}_j^1\right)$.
4. Finally P_s sends the set $\left\{\bar{\boldsymbol{c}}_j\right\}_{j\in\mathcal{J}}$ to P_r. For $l\in[n-k]$ if $b_{k+l}=1$, P_r updates $w_j^{k+l}:=\bar{\boldsymbol{c}}_j[l]+w_j^{k+l}$.

Consistency Check [a]

5. P_r samples uniformly random values $(x_1,...,x_\gamma)\in_R\mathbb{F}^\gamma$ and sends these to P_s.
6. P_s then computes

$$\tilde{\boldsymbol{r}}^0=\boldsymbol{r}^0_{\mathcal{T}+\gamma+1}+\sum_{j=1}^{\gamma}x_j\boldsymbol{r}^0_{\mathcal{T}+j},\quad \tilde{\boldsymbol{r}}^1=\boldsymbol{r}^1_{\mathcal{T}+\gamma+1}+\sum_{j=1}^{\gamma}x_j\boldsymbol{r}^1_{\mathcal{T}+j},\quad \tilde{\boldsymbol{c}}^0=\boldsymbol{c}^0_{\mathcal{T}+\gamma+1}+\sum_{j=1}^{\gamma}x_j\boldsymbol{c}^0_{\mathcal{T}+j}$$

   and sends $\left(\tilde{\boldsymbol{r}}^0,\tilde{\boldsymbol{r}}^1,\tilde{\boldsymbol{c}}^0\right)$ to P_r.
7. P_r computes $\tilde{\boldsymbol{w}}=\boldsymbol{w}_{\mathcal{T}+\gamma+1}+\sum_{j=1}^{\gamma}x_j\boldsymbol{w}_{\mathcal{T}+j}$ and $\tilde{\boldsymbol{t}}\leftarrow\mathcal{C}\left(\tilde{\boldsymbol{r}}^0+\tilde{\boldsymbol{r}}^1\right)$. It lets $\tilde{\boldsymbol{c}}\leftarrow\pi_{k+1,n}\left(\tilde{\boldsymbol{t}}\right)$ and lets $\tilde{\boldsymbol{c}}^1=\tilde{\boldsymbol{c}}-\tilde{\boldsymbol{c}}^0$. Finally for $u\in[k]$ and $v\in[n-k]$, P_r verifies that $\tilde{\boldsymbol{r}}^{b_u}[u]=\tilde{\boldsymbol{w}}[u]$ and $\tilde{\boldsymbol{c}}^{b_{k+v}}[v]=\tilde{\boldsymbol{w}}[k+v]$. If this is not the case then P_r output abort and halts.

Output

8. Both parties increment their local counter $\mathcal{T}:=\mathcal{T}+\gamma$. P_s now holds opening information $\left\{\left(\boldsymbol{r}_j^0,\boldsymbol{r}_j^1,\boldsymbol{c}_j^0\right)\right\}_{j\in[\mathcal{T}]}$ and P_r holds the verifying information $\left\{\boldsymbol{w}_j\right\}_{j\in[\mathcal{T}]}$. Let $\overline{\mathcal{J}}=\mathcal{J}\setminus\{\mathcal{T}+\gamma+1\}$. P_s outputs $\left(\texttt{committed},\texttt{sid},\overline{\mathcal{J}},\{\boldsymbol{r}_j\}_{j\in\overline{\mathcal{J}}}\right)$ and P_r outputs $\left(\texttt{receipt},\texttt{sid},\overline{\mathcal{J}}\right)$.[b]

[a] This test assumes that $|\mathbb{F}|^{-1}\leq 2^{-s}$. If this is not the case, the test will be repeated a number $\hat{s}$ times such that $|\mathbb{F}|^{-\hat{s}}\leq 2^{-s}$. This requires generating a factor $\hat{s}$ additional random field elements.

[b] For clarity we assume that P_s and P_r locally discard the extra commitment used for blinding. This includes the bookkeeping of the index offset this creates when generating multiple batches.

Fig. 4. Protocol Π_{HCOM} UC-realizing $\mathcal{F}_{\mathsf{HCOM}}$ in the $\mathcal{F}_{\mathsf{ROT}}$-hybrid model – part 1.

However the main advantage of our approach comes from ensuring that the shares held by P_r binds the sender P_s to his committed value, while only requiring a minimum distance of s. On top of that our approach is also additively homomorphic. The idea is that P_r will challenge P_s to open a random linear combination of all the committed values and check that these are valid according to $\mathcal{C}$. Recall that $\gamma+1$ commitments are produced in total. The reason for this is to guarantee hiding for the commitments, even when P_r learns a random linear combination of them. Therefore, the linear combination is "blinded" by a pseudorandom value only used once and thus it appears pseudorandom to P_r as well. This approach requires setting up commitments to 1 extra value for each invocation of **Commit**. This consistency check is sufficient if $|\mathbb{F}|^{-1} \leq 2^{-s}$, however if the field is too small then the check is simply repeated $\hat{s} = \lceil s/\log_2(|\mathbb{F}|) \rceil$ times such that $|\mathbb{F}|^{-\hat{s}} \leq 2^{-s}$.

Open:

1. On input $\left(\texttt{open},\texttt{sid},\{(c,\alpha_c)\}_{c\in C}\right)$ where each $\alpha_c \in \mathbb{F}$, if for all $c \in C$, P_s holds $\left(\boldsymbol{r}_c^0,\boldsymbol{r}_c^1,\boldsymbol{c}_c^0\right)$ it computes
$$\boldsymbol{r}^0 = \sum_{c\in C} \alpha_c \cdot \boldsymbol{r}_c^0, \quad \boldsymbol{r}^1 = \sum_{c\in C} \alpha_c \cdot \boldsymbol{r}_c^1, \quad \boldsymbol{c}^0 = \sum_{c\in C} \alpha_c \cdot \boldsymbol{c}_c^0$$
and sends $\left(\texttt{opening},\{c,\alpha_c\}_{c\in C},\left(\boldsymbol{r}^0,\boldsymbol{r}^1,\boldsymbol{c}^0\right)\right)$ to P_r. Else it ignores the input message.
2. Upon receiving the message $\left(\texttt{opening},\{c,\alpha_c\}_{c\in C},\left(\boldsymbol{r}^0,\boldsymbol{r}^1,\boldsymbol{c}^0\right)\right)$ from P_s, if for all $c \in C$, P_r holds $\boldsymbol{w}_c$ it lets $\boldsymbol{r} = \boldsymbol{r}^0 + \boldsymbol{r}^1$ and computes $\boldsymbol{w} = \sum_{c\in C} \alpha_c \cdot \boldsymbol{w}_c$ and $\boldsymbol{t} \leftarrow \mathcal{C}(\boldsymbol{r})$. It lets $\boldsymbol{c} = \pi_{k+1,n}(\boldsymbol{t})$ and computes $\boldsymbol{c}^1 = \boldsymbol{c} - \boldsymbol{c}^0$. Finally for $i \in [k]$ and $l \in [n-k]$, P_r verifies that $\boldsymbol{r}^{b_i}[i] = \boldsymbol{w}[i]$ and $\boldsymbol{c}^{b_{k+l}}[l] = \boldsymbol{w}[k+l]$. If all checks are valid P_r outputs $\left(\texttt{opened},\texttt{sid},\{(c,\alpha_c)\}_{c\in C},\boldsymbol{r}\right)$. Else it aborts and halts.

Fig. 5. Protocol Π_{HCOM} UC-realizing $\mathcal{F}_{\mathsf{HCOM}}$ in the $\mathcal{F}_{\mathsf{ROT}}$-hybrid model – part 2.

The intuition why the above approach works is that if the sender P_s sent inconsistent corrections, then it will get challenged on these positions with high probability. In order to pass the check, P_s must therefore guess which choice-bit P_r holds for each position for which it sent inconsistent values. The challenge therefore forces P_s to make a decision at commitment time which underlying value to send consistent openings to, and after that it can only open to that value successfully. In fact, the above approach also guarantees that the scheme is homomorphic. This is because all the freedom P_s might have had by sending maliciously constructed corrections is removed already at commitment time, so after this phase commitments and shares can be added together without issue.

To extract *all* committed values when receiving the opening to the linear combination the simulator identifies which rows of $\boldsymbol{S}^0$ and $\boldsymbol{S}^1$ P_s is sending inconsistent shares for. For these positions it inserts erasures in all positions of $\boldsymbol{t}_j$ (as defined by $\boldsymbol{S}^0, \boldsymbol{S}^1, \tilde{\boldsymbol{c}}_j$ and $\mathcal{C}$). As there are at most $s-1$ positions where P_s could have cheated and the distance of the *linear* code is $d \geq s$ the simulator can erasure decode all columns to a unique value, and this is the only value P_s can successfully open to.[1]

2.3 Protocol Extension

The protocol Π_{HCOM} implements a commitment scheme where the sender commits to pseudorandom values. In many applications however it is needed to commit to chosen values instead. It is know that for any UC-secure commitment scheme one can easily turn a commitment from a random value into a commitment of a chosen one using the random value as a one-time pad encryption of the chosen value. For completeness, in Appendix A, we show this extension for any protocol implementing $\mathcal{F}_{\mathsf{HCOM}}$.

In addition we also highlight that all additively homomorphic commitment schemes support the notion of batch-opening. For applications where a large amount of messages need to be opened at the same time this has great implications on efficiency. The technique is roughly that P_s sends the values he wants to open to directly to P_r. To verify correctness the receiver then challenges the sender to open to a random linear combination of the received messages. For the same reason as for the consistency check of **Commit** this optimization retains binding. Using this method the overhead of opening the commitments is independent of the number of messages opened to in a batch and therefore amortizes away in the same manner as the consistency check and the initial OTs. However this way of opening messages has the downside of making the opening phase interactive, which is not optimal for all applications. See Appendix A for details.

[1] All linear codes can be efficiently erasure decoded if the number of erasures is $\leq d-1$.

The abovementioned batch-opening technique also has applicability when committing to large messages. Say we want to commit to a message m of length M. The naive approach would be to instantiate our scheme using a $[n_M, M, s]$ code. However this would require $n_M \geq M$ initial OTs and in addition only achieve rate $M/(M+n_M) \geq 1/2$ in the opening phase. Instead of the above, the idea is to break the large message of length M into blocks of length l for $l << M$. There will now be $N = \lceil M/l \rceil$ of these blocks in total. We then instantiate our scheme with a $[n_s, l, s]$ code and commit to m in blocks of size l. When required to open we use the above-mentioned batch-opening to open all blocks of m. It is clear that the above technique remains additively homomorphic for commitments to the large messages. In [GIKW14] they show an example for messages of size 2^{30} where they achieve rate $1.046^{-1} \approx 0.95$ in both the commit and open phase. In Appendix A we apply our above approach to the same setting and conclude that in the commit phase we achieve rate ≈ 0.974 and even higher in the opening phase. This is including the cost of the initial OTs.

3 Security

In this section we prove the following theorem.

Theorem 1. *The protocol Π_{HCOM} in Fig. 4 and Fig. 5 UC-realizes the $\mathcal{F}_{\mathsf{HCOM}}$ functionality of Fig. 2 in the $\mathcal{F}_{\mathsf{ROT}}$-hybrid model against any number of static corruptions.*

Proof. We prove security for the case with a dummy adversary, so that the simulator is outputting simulated values directly to the environment and is receiving inputs directly from the environment. We focus on the case with one call to **Commit**. The proof trivially lifts to the case with multiple invocations. The case with two static corruptions is trivial. The case with no corruptions follows from the case with a corrupted receiver, as in the ideal functionality $\mathcal{F}_{\mathsf{HCOM}}$ the adversary is given all values which are given to the receiver, so one can just simulate the corrupted receiver and then output only the public transcript of the communication to the environment. We now first prove the case with a corrupted receiver and then the case with a corrupted sender.

Assume that P_r is corrupted. We use $\breve{P}_r$ to denote the corrupted receiver. This is just a mnemonic pseudonym for the environment $\mathcal{Z}$. The main idea behind the simulation is to simply run honestly until the opening phase. In the opening phase we then equivocate the commitment to the value received from the ideal functionality $\mathcal{F}_{\mathsf{HCOM}}$ by adjusting the bits $\bar{\boldsymbol{s}}_j^{1-b_i}$ not being watched by the receiver. This will be indistinguishable from the real world as the vectors $\bar{\boldsymbol{s}}_i^{1-b_i}$ are indistinguishable from uniform in the view of $\breve{P}_r$ and if all the vectors $\bar{\boldsymbol{s}}_i^{1-b_i}$ were uniform, then adjusting the bits not watched by $\breve{P}_r$ would be perfectly indistinguishable.

We first describe how to simulate the protocol without the step *Consistency Check*. We then discuss how to extend the simulation to this case.

The simulator $\mathcal{S}$ will run **Init** honestly, simulating $\mathcal{F}_{\mathsf{ROT}}$ to $\breve{P}_r$. It then runs **Commit** honestly. On input $(\texttt{opened}, \texttt{sid}, \{(c, \alpha_c)\}_{c \in C}, \boldsymbol{r})$ it must simulate an opening.

In the simulation we use the fact that in the real protocol P_r can recompute all the values received from P_s given just the value $\boldsymbol{r}$ and the values $\boldsymbol{w}_c$, which it already knows, and assuming that the checks $\boldsymbol{r}^{b_i}[i] = \boldsymbol{w}[i]$ and $\boldsymbol{c}^{b_{k+l}}[l] = \boldsymbol{w}[k+l]$ at the end of Fig. 5 are true. This goes as follows: First compute $\boldsymbol{w} = \sum_{c \in C} \alpha_c \cdot \boldsymbol{w}_c$, $\boldsymbol{t} = \mathcal{C}(\boldsymbol{r})$ and $\boldsymbol{c} = \pi_{k+1,n}(\boldsymbol{t})$, as in the protocol. Then for $i \in [k]$ and $l \in [n-k]$ define

$$\boldsymbol{r}^{b_i}[i] = \boldsymbol{w}[i] \ , \ \boldsymbol{c}^{b_{k+l}}[l] = \boldsymbol{w}[k+l] \ . \tag{1}$$

$$\boldsymbol{r}^{1-b_i}[i] = \boldsymbol{r}[i] - \boldsymbol{r}^{b_i}[i] \ , \ \boldsymbol{c}^{1-b_{k+l}}[l] = \boldsymbol{c}[l] - \boldsymbol{c}^{b_{k+l}}[l] \ . \tag{2}$$

In (1) we use that the checks are true. In (2) we use that $\boldsymbol{r}=\boldsymbol{r}^0+\boldsymbol{r}^1$ and $\boldsymbol{c}^1=\boldsymbol{c}-\boldsymbol{c}^0$ by construction of P_r. This clearly correctly recomputes $(\boldsymbol{r}^0,\boldsymbol{r}^1,\boldsymbol{c}^0)$.

On input $(\mathtt{opened},\mathtt{sid},\{(c,\alpha_c)\}_{c\in C},\boldsymbol{r})$ from $\mathcal{F}_{\mathsf{HCOM}}$, the simulator will compute $(\boldsymbol{r}^0,\boldsymbol{r}^1,\boldsymbol{c}^0)$ from $\boldsymbol{r}$ and the values $\boldsymbol{w}_c$ known by $\breve{P}_r$ as above and send $(\mathtt{opening},\{c,\alpha_c\}_{c\in C},(\boldsymbol{r}^0,\boldsymbol{r}^1,\boldsymbol{c}^0))$ to $\breve{P}_r$.

We now argue that the simulation is computationally indistinguishable from the real protocol. We go via two hybrids.

We define *Hybrid I* as follows. Instead of computing the rows $\bar{\mathbf{s}}_i^{1-b_i}$ from the seeds $l_i^{1-b_i}$ the simulator samples $\bar{\mathbf{s}}_i^{1-b_i}$ uniformly at random of the same length. Since $\breve{P}_r$ never sees the seeds $l_i^{1-b_i}$ and P_s only uses them as input to PRG, we can show that the view of $\breve{P}_r$ in the simulation and Hybrid I are computationally indistinguishable by a black box reduction to the security of PRG.

We define *Hybrid II* as follows. We start from the real protocol, but instead of computing the rows $\bar{\mathbf{s}}_i^{1-b_i}$ from the seeds $l_i^{1-b_i}$ we again sample $\bar{\mathbf{s}}_i^{1-b_i}$ uniformly at random of the same length. As above, we can show that the view of $\breve{P}_r$ in the protocol and Hybrid II are computationally indistinguishable.

The proof then concludes by transitivity of computational indistinguishability and by observing that the views of $\breve{P}_r$ in Hybrid I and Hybrid II are perfectly indistinguishable. The main observation needed for seeing this is that in Hybrid I all the bits $\boldsymbol{r}_j[i]$ are chosen uniformly at random and independently by $\mathcal{F}_{\mathsf{HCOM}}$, whereas in Hybrid II they are defined by $\boldsymbol{r}_j[i]=\boldsymbol{r}_j^0[i]+\boldsymbol{r}_j^1[i]=\boldsymbol{r}_j^{b_i}[i]+\boldsymbol{r}_j^{1-b_i}[i]$, where all the bits $\boldsymbol{r}_j^{1-b_i}[i]$ are chosen uniformly at random and independently by $\mathcal{S}$. This yields the same distributions of the values $\boldsymbol{r}_j$. All other value clearly have the same distribution.

We now address the step *Consistency Check*. The simulation of this step follows the same pattern as above. Define $\tilde{\boldsymbol{r}}=\tilde{\boldsymbol{r}}^0+\tilde{\boldsymbol{r}}^1$. This is the value from which $\tilde{\boldsymbol{t}}$ is computed in Step 7 in Fig. 4. In the simulation and Hybrid I, instead pick $\tilde{\boldsymbol{r}}$ uniformly at random and then recompute the values sent to $\breve{P}_r$ as above. In Hybrid II compute $\tilde{\boldsymbol{r}}$ as in the protocol (but still starting from the uniformly random $\bar{\mathbf{s}}_i^{1-b_i}$). Then simply observe that $\tilde{\boldsymbol{r}}$ has the same distribution in Hybrid I and Hybrid II. In Hybrid I it is uniformly random. In Hybrid II it is computed as $\tilde{\boldsymbol{r}}^0+\tilde{\boldsymbol{r}}^1=(\boldsymbol{r}^0_{\mathcal{T}+\gamma+1}+\boldsymbol{r}^1_{\mathcal{T}+\gamma+1})+\sum_{j=1}^{\gamma}x_j\boldsymbol{r}_{\mathcal{T}+j}$, and it is easy to see that $\boldsymbol{r}^0_{\mathcal{T}+\gamma+1}+\boldsymbol{r}^1_{\mathcal{T}+\gamma+1}$ is uniformly random and independent of all other values in the view of $\breve{P}_r$.

We now consider the case where the sender is corrupted who we denote $\breve{P}_s$. The simulator will run the code of P_s honestly, simulating also $\mathcal{F}_{\mathsf{ROT}}$ honestly. It will record the values (b_i,l_i^0,l_i^1) from **Init**. The remaining job of the simulator is then to extract the values $\tilde{\boldsymbol{r}}_j$ to send to $\mathcal{F}_{\mathsf{HCOM}}$ in the command $\left(\mathtt{corrupt\text{-}commit},\mathtt{sid},\{\tilde{\boldsymbol{r}}_j\}_{j\in\mathcal{J}}\right)$. This should be done such that the probability that the receiver later outputs $(\mathtt{opened},\mathtt{sid},\{(c,\alpha_c)\}_{c\in C},\boldsymbol{r})$ for $\boldsymbol{r}\neq\sum_{c\in C}\alpha_c\tilde{\boldsymbol{r}}_c$ is at most 2^{-s}. We first describe how to extract the values $\tilde{\boldsymbol{r}}_j$ and then show that the commitments are binding to these values.

We use the *Consistency Check* performed in the second half of Fig. 4 to define a set $E\subseteq\{1,\ldots,n\}$. We call this the erasure set. This name will make sense later, but for now think of E as the set of indices for which the corrupted sender $\breve{P}_s$ after the consistency checks knows the choice bits b_i for $i\in E$ and for which the bits b_i for $i\notin E$ are still uniform in the view of $\breve{P}_s$.

Define the column vectors $\boldsymbol{s}_j^0$ and $\boldsymbol{s}_j^1$ as in the protocol. This is possible as the seeds from $\mathcal{F}_{\mathsf{ROT}}$ are well defined. Following the protocol, and adding a few more definitions, define

$$\boldsymbol{r}_j^0=\pi_k(\boldsymbol{s}_j^0)\ ,\ \boldsymbol{r}_j^1=\pi_k(\boldsymbol{s}_j^1)\ ,\ \boldsymbol{r}_j=\boldsymbol{r}_j^0+\boldsymbol{r}_j^1\ ,\ \boldsymbol{u}_j^0=\pi_{k+1,n}(\boldsymbol{s}_j^0)\ ,\ \boldsymbol{u}_j^1=\pi_{k+1,n}(\boldsymbol{s}_j^1)\ ,\ \boldsymbol{u}_j=\boldsymbol{u}_j^0+\boldsymbol{u}_j^1$$
$$\boldsymbol{t}_j=\mathcal{C}(\boldsymbol{r}_j)\ ,\ \boldsymbol{c}_j=\pi_{k+1,n}(\boldsymbol{t}_j)\ ,\ \boldsymbol{c}_j^0=\boldsymbol{u}_j^0\ ,\ \boldsymbol{c}_j^1=\boldsymbol{c}_j-\boldsymbol{c}_j^0$$
$$\boldsymbol{d}_j^0=\boldsymbol{u}_j^0\ ,\ \boldsymbol{d}_j^1=\boldsymbol{u}_j^1+\bar{\boldsymbol{c}}_j\ ,\ \boldsymbol{d}_j=\boldsymbol{d}_j^0+\boldsymbol{d}_j^1=\boldsymbol{u}_j+\bar{\boldsymbol{c}}_j\ ,\ \boldsymbol{w}_j^0=\boldsymbol{r}_j^0\|\boldsymbol{d}_j^0\ ,\ \boldsymbol{w}_j^1=\boldsymbol{r}_j^1\|\boldsymbol{d}_j^1\ .$$

Notice that if P_s is honest, then $\bar{\boldsymbol{c}}_j=\boldsymbol{c}_j-\boldsymbol{u}_j$ and therefore $\boldsymbol{d}_j=\boldsymbol{d}_j^0+\boldsymbol{d}_j^1=\boldsymbol{u}_j^0+\boldsymbol{u}_j^1+\bar{\boldsymbol{c}}_j=\boldsymbol{c}_j$. Hence $\boldsymbol{d}_j^0$ and $\boldsymbol{d}_j^1$ are the two shares of the non-systematic part $\boldsymbol{c}_j$ the same way that $\boldsymbol{r}_j^0$ and $\boldsymbol{r}_j^1$ are the two shares of the systematic part $\boldsymbol{r}_j$. If the sender was honest we would in particular have that $\boldsymbol{w}_j^0+\boldsymbol{w}_j^1=\boldsymbol{r}_j\|\boldsymbol{d}_j=\boldsymbol{r}_j\|\boldsymbol{c}_j=\mathcal{C}(\boldsymbol{r}_j)$, *i.e.*, $\boldsymbol{w}_j^0$ and $\boldsymbol{w}_j^1$ would be the two shares of the whole codeword.

We can define the values that an honest P_s *should* send as

$$\tilde{\boldsymbol{r}}^0 = \boldsymbol{r}^0_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{r}^0_j \ , \ \tilde{\boldsymbol{r}}^1 = \boldsymbol{r}^1_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{r}^1_j \ , \ \tilde{\boldsymbol{c}}^0 = \boldsymbol{c}^0_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{c}^0_j \ .$$

These values can be used to define values

$$\tilde{\boldsymbol{r}} = \tilde{\boldsymbol{r}}^0 + \tilde{\boldsymbol{r}}^1 \ , \ \tilde{\boldsymbol{t}} = \mathcal{C}(\tilde{\boldsymbol{r}}) \ , \ \tilde{\boldsymbol{c}} = \pi_{k+1,n}(\tilde{\boldsymbol{t}}) \ , \ \tilde{\boldsymbol{c}}^1 = \tilde{\boldsymbol{c}} - \tilde{\boldsymbol{c}}^0 \ , \ \tilde{\boldsymbol{w}}^0 = \tilde{\boldsymbol{r}}^0 \| \tilde{\boldsymbol{c}}^0 \ , \ \tilde{\boldsymbol{w}}^1 = \tilde{\boldsymbol{r}}^1 \| \tilde{\boldsymbol{c}}^1 \ .$$

We use $(\breve{\boldsymbol{r}}^0, \breve{\boldsymbol{r}}^1, \breve{\boldsymbol{c}}^0)$ to denote the values actually sent by $\breve{P}_s$ and we let the following denote the values computed by P_r (plus some extra definitions).

$$\breve{\boldsymbol{r}} = \breve{\boldsymbol{r}}^0 + \breve{\boldsymbol{r}}^1 \ , \ \breve{\boldsymbol{t}} = \mathcal{C}(\breve{\boldsymbol{r}}) \ , \ \breve{\boldsymbol{c}} = \pi_{k+1,n}(\breve{\boldsymbol{t}}) \ , \ \breve{\boldsymbol{c}}^1 = \breve{\boldsymbol{c}} - \breve{\boldsymbol{c}}^0 \ , \ \breve{\boldsymbol{w}}^0 = \breve{\boldsymbol{r}}^0 \| \breve{\boldsymbol{c}}^0 \ , \ \breve{\boldsymbol{w}}^1 = \breve{\boldsymbol{r}}^1 \| \breve{\boldsymbol{c}}^1 \ , \ \breve{\boldsymbol{w}} = \breve{\boldsymbol{w}}^0 + \breve{\boldsymbol{w}}^1 \ .$$

The simulator computes

$$\tilde{\boldsymbol{w}} = \boldsymbol{w}_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{w}_{\mathcal{T}+j} \tag{3}$$

as P_r in the protocol. For later use, define $\tilde{\boldsymbol{w}}^0 = \boldsymbol{w}^0_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{w}^0_{\mathcal{T}+j}$ and $\tilde{\boldsymbol{w}}^1 = \boldsymbol{w}^1_{\mathcal{T}+\gamma+1} + \sum_j x_j \boldsymbol{w}^1_{\mathcal{T}+j}$. The check performed by P_r is then simply to check for $u = 1,...,n$ that

$$\breve{\boldsymbol{w}}^{b_u}[u] = \tilde{\boldsymbol{w}}[u] \ . \tag{4}$$

Notice that in the protocol we have that $\boldsymbol{w}_j = \boldsymbol{b} * (\boldsymbol{w}^1_j - \boldsymbol{w}^0_j) + \boldsymbol{w}^0_j$, where $*$ denotes the Schur product also known as the positionwise product of vectors. To see this notice that $(\boldsymbol{b} * (\boldsymbol{w}^1_j - \boldsymbol{w}^0_j) + \boldsymbol{w}^0_j)[i] = b_i(\boldsymbol{w}^1_j[i] - \boldsymbol{w}^0_j[i]) + \boldsymbol{w}^0_j[i] = \boldsymbol{w}^{b_i}_j[i]$. In other words, $\boldsymbol{w}_j[i] = \boldsymbol{w}^{b_i}_j[i]$. It then follows from (3) that $\tilde{\boldsymbol{w}} = \boldsymbol{b} * (\tilde{\boldsymbol{w}}^1 - \tilde{\boldsymbol{w}}^0) + \tilde{\boldsymbol{w}}^0$, from which it follows that $\tilde{\boldsymbol{w}}[u] = \tilde{\boldsymbol{w}}^{b_u}[u]$. From (4) it then follows that $\breve{P}_s$ passes the consistency check if and only if for $u = 1,...,n$ it holds that $\breve{\boldsymbol{w}}^{b_u}[u] = \tilde{\boldsymbol{w}}^{b_u}[u]$. We make some definitions related to this check. We say that a position $u \in [n]$ is *silly* if $\breve{\boldsymbol{w}}^0[u] \neq \tilde{\boldsymbol{w}}^0[u]$ and $\breve{\boldsymbol{w}}^1[u] \neq \tilde{\boldsymbol{w}}^1[u]$. We say that a position $u \in [n]$ is *clean* if $\breve{\boldsymbol{w}}^0[u] = \tilde{\boldsymbol{w}}^0[u]$ and $\breve{\boldsymbol{w}}^1[u] = \tilde{\boldsymbol{w}}^1[u]$. We say that a position $u \in [n]$ is *probing* if it is not silly or clean. Let E denote the set of probing positions u. Notice that if there is a silly position u, then $\breve{\boldsymbol{w}}^{b_u}[u] \neq \tilde{\boldsymbol{w}}^{b_u}[u]$ so $\breve{P}_s$ gets caught. We can therefore assume without loss of generality that there are no silly positions. For the probing positions $u \in E$, there is by definition a bit c_u such that $\breve{\boldsymbol{w}}^{1-c_u}[u] \neq \tilde{\boldsymbol{w}}^{1-c_u}[u]$ and such that $\breve{\boldsymbol{w}}^{c_u}[u] = \tilde{\boldsymbol{w}}^{c_u}[u]$. This means that $\breve{P}_s$ passes the test only if $c_u = b_u$ for all $u \in E$. Since $\breve{P}_s$ knows c_u it follows that if $\breve{P}_s$ does not get caught, then it can guess b_u for $u \in E$ with probability 1.

Before we proceed to describe the extractor, we are now going to show two facts about E. First we will show that $|E| < s$, except with probability 2^{-s}. This follows from the simple observation that each b_u for $u \in E$ is uniformly random and $\breve{P}_s$ passes the consistency test if and only if $c_u = b_u$ for $u \in E$ and the only information that $\breve{P}_s$ has on the bits b_u is via the probing positions. Hence $\breve{P}_s$ passes the consistency test with probability at most $2^{-|E|}$.

Second, let $\mathcal{C}_{-E}$ be the code obtained from $\mathcal{C}$ by puncturing at the positions $u \in E$, *i.e.*, a codeword of $\mathcal{C}_{-E}$ can be computed as $\boldsymbol{t} = \mathcal{C}(\boldsymbol{r})$ and then outputting $\boldsymbol{t}_{-E}$, *i.e.*, the vector $\boldsymbol{t}$ where we remove the positions $u \in E$. We show that for all $j = \mathcal{T}+1,...,\mathcal{T}+\gamma$ it holds that $(\boldsymbol{w}^0_j + \boldsymbol{w}^1_j)_{-E} \in \mathcal{C}_{-E}(\mathbb{F}^k)$, except with probability 2^{-s}. To see this, assume for the sake of contradiction that there exists such j where $(\boldsymbol{w}^0_j + \boldsymbol{w}^1_j)_{-E} \notin \mathcal{C}_{-E}(\mathbb{F}^k)$. Then the probability that it does not happen that $(\tilde{\boldsymbol{w}}^0 + \tilde{\boldsymbol{w}}^1)_{-E} \notin \mathcal{C}_{-E}(\mathbb{F}^k)$ is at most $|\mathbb{F}|^{-1}$, as a random linear combination of non-codewords become a codeword with probability at most $|\mathbb{F}|^{-1}$.[2] We repeat this test a number $\hat{s}$ of times such that $|\mathbb{F}|^{-\hat{s}} \leq 2^{-s}$. Since the tests succeed independently with probability at most $|\mathbb{F}|^{-1}$ it follows that $(\tilde{\boldsymbol{w}}^0 + \tilde{\boldsymbol{w}}^1)_{-E} \notin \mathcal{C}_{-E}(\mathbb{F}^k)$ except with probability 2^{-s}. Since by construction $\breve{\boldsymbol{w}}^0 + \breve{\boldsymbol{w}}^1 \in \mathcal{C}(\mathbb{F}^k)$, we have that $(\breve{\boldsymbol{w}}^0 + \breve{\boldsymbol{w}}^1)_{-E} \in \mathcal{C}_{-E}(\mathbb{F}^k)$, so when $(\tilde{\boldsymbol{w}}^0 +$

[2] In it easy to see that in general, a random linear combination of vectors from outside some linear subspace will end up in the subspace with probability at most $|\mathbb{F}|^{-1}$.

$\tilde{\boldsymbol{w}}^1)_{-E}\notin\mathcal{C}_{-E}(\mathbb{F}^k)$ we either have that $(\tilde{\boldsymbol{w}}^0)_{-E}\neq(\breve{\boldsymbol{w}}^0)_{-E}$ or $(\tilde{\boldsymbol{w}}^1)_{-E}\neq(\breve{\boldsymbol{w}}^1)_{-E}$. Since there are no silly positions, this implies that we have a new probing position $u\notin E$, a contradiction to the definition of E.

We can now assume without loss of generality that $|E|<s$ and that $(\boldsymbol{w}_j^0+\boldsymbol{w}_j^1)_{-E}\in\mathcal{C}_{-E}(\mathbb{F}^k)$. From $|E|<s$ and $\mathcal{C}$ having minimal distance $d\geq s$ we have that $\mathcal{C}_{-E}$ has minimal distance ≥ 1. Hence we can from each j and each $(\boldsymbol{w}_j^0+\boldsymbol{w}_j^1)_{-E}\in\mathcal{C}_{-E}(\mathbb{F}^k)$ compute $\tilde{\boldsymbol{r}}_j\in\mathbb{F}^k$ such that $(\boldsymbol{w}_j^0+\boldsymbol{w}_j^1)_{-E}=\mathcal{C}_{-E}(\tilde{\boldsymbol{r}}_j)$. These are the values that $\mathcal{S}$ will send to $\mathcal{F}_{\mathsf{HCOM}}$.

We then proceed to show that for all $\{(c,\alpha_c)\}_{c\in C}$ the environment can open to $(\texttt{opened},\texttt{sid},\{(c,\alpha_c)\}_{c\in C},\tilde{\boldsymbol{r}})$ for $\tilde{\boldsymbol{r}}=\sum_{c\in C}\alpha_c\tilde{\boldsymbol{r}}_c$ with probability 1. The reason for this is that if $\breve{P}_s$ computes the values in the opening correctly, then clearly $(\breve{\boldsymbol{w}}^0)_{-E}=(\tilde{\boldsymbol{w}}^0)_{-E}$ and $(\breve{\boldsymbol{w}}^1)_{-E}=(\tilde{\boldsymbol{w}}^1)_{-E}$. Furthermore, for the positions $u\in E$ it can open to any value as it knows b_u. It therefore follows that if $\breve{P}_s$ can open to $(\texttt{opened},\texttt{sid},\{(c,\alpha_c)\}_{c\in C},\boldsymbol{r})$ for $\boldsymbol{r}\neq\sum_{c\in C}\alpha_c\tilde{\boldsymbol{r}}_c$, then it can open $\{(c,\alpha_c)\}_{c\in C}$ to two different values. Since the code has distance $d\geq s$, it is easy to see that after opening some $\{(c,\alpha_c)\}_{c\in C}$ to two different values, the environment can compute with probability 1 at least s of the choice bits b_u, which it can do with probability at most 2^{-s}, which is negligible. □

4 Efficiency Comparison

In this section we compare the efficiency of our scheme to the most efficient schemes in the literature realizing UC-secure commitments with security against a static and malicious adversary. In particular, we compare our construction to the schemes of [Lin11], [BCPV13] and [CDD+15].

The scheme of [BCPV13] (Fig. 6) is a slightly optimized version of [Lin11] (Protocol 2) which implement a multi-commitment ideal functionality. This means these schemes support commitments to multiple parties natively, a property not shared with the rest of the protocols in this comparison. We therefore only consider the two party case where a sender commits to a receiver. Both schemes are cast in the CRS-model and their security relies on the DDH assumption. As the messages to be committed to are encoded as group elements the message size and the level of security are coupled in these schemes. For big messages this is not really an issue as the group size would just increase as well. However, for small messages, it is not possible to decrease the group size, as this would weaken security. The authors propose instantiating their scheme over an elliptic curve group over a field size of 256-bits so later in our comparison we also consider committing to values of this length. This is optimal for these schemes as the overhead of working with group elements of 256-bits would become more apparent if committing to smaller values.

Scheme	Homo	OTs		Communication		Rounds		Computation			
		$\binom{2}{1}$	$\binom{3}{2}$	Commit	Open	Commit	Open	Commit		Open	
								Exp.	Enc.	Exp.	Enc.
[Lin11]	✗	0	0	$4g$	$6g+4l+k$	1	5	5	0	$18\frac{1}{3}$	0
[BCPV13]	✗	0	0	$4g$	$5g+3l+k$	1	3	10	0	12	0
[CDD+15], basic	✗	$\overline{n}$	0	$2\overline{n}f$	$(k+\overline{n}+1)f$	1	1	0	1	0	1
[CDD+15]	✓	0	$\overline{n}$	$^{6(k+2\overline{n})\overline{n}f}/_{k}+kf$	$(k+2\overline{n}+1)f$	1	1	0	$^{8\overline{n}}/_{k}+2$	0	1
This Work	✓	n	0	$^{(2\hat{s}nf+\kappa)}/_{\gamma}+nf$	$(k+n+1)f$	3	1	0	$^{2\hat{s}}/_{\gamma}+1$	0	1

Table 1. Comparison of the most efficient UC-secure schemes for committing to γ messages of k components. Sizes are in bits. Legend: g is size of a group element, l is size of a scalar in the exponent, f is the size of a finite field element, $\hat{s}$ is the number of consistency checks performed, Exp. denotes the number of modular exponentiations, Enc. denotes the number of encoding procedures of the corresponding codes which have length $\overline{n}$ and n. The schemes of [CDD+15] are presented with the sharing parameter t set to 2 for the basic and 3 for the homomorphic.

We present our detailed comparison in Table 1. The table shows the costs of all the previously mentioned schemes in terms of OTs required, communication, number of rounds and computation. For the schemes of [CDD+15] we have fixed the sharing parameter t to 2 and 3 for the basic and homomorphic version, respectively. To the best of our knowledge this is also the optimal choice in all settings. For our scheme, instead of counting the cost of sending the challenges $(x_1,x_2,\ldots,x_\gamma) \in \mathbb{F}$, we assume the receiver sends a random seed of size κ instead. This is then used as input to a PRG whose output is used to determine the challenges.

Scheme	Homo	OTs		Communication		Rounds		Computation			
		$\binom{2}{1}$	$\binom{3}{2}$	Commit	Open	Commit	Open	Commit		Open	
								Exp.	Enc.	Exp.	Enc.
[Lin11]	✗	0	0	1024	2816	1	5	5	0	$18\frac{1}{3}$	0
[BCPV13]	✗	0	0	1024	2304	1	3	10	0	12	0
[CDD$^+$15], basic, $\gamma = 304$	✗	631	0	4593	888	1	1	24	1	0	1
[CDD$^+$15], $\gamma = 304$	✓	0	631	37138	1519	1	1	96	22	0	1
This Work, $\gamma = 304$	✓	419	0	2647	676	3	1	16	1.3	0	1
[CDD$^+$15], basic, $\gamma = 1000$	✗	631	0	2232	888	1	1	7	1	0	1
[CDD$^+$15], $\gamma = 1000$	✓	0	631	26905	1519	1	1	28	22	0	1
This Work, $\gamma = 1000$	✓	419	0	1097	676	3	1	5	1	0	1
[CDD$^+$15], basic, $\gamma = 10000$	✗	631	0	1359	888	1	1	0	1	0	1
[CDD$^+$15], $\gamma = 10000$	✓	0	631	23125	1519	1	1	3	22	0	1
This Work, $\gamma = 10000$	✓	419	0	487	676	3	1	0	1	0	1
[CDD$^+$15], basic, $\gamma = 100000$	✗	631	0	1271	888	1	1	0	1	0	1
[CDD$^+$15], $\gamma = 100000$	✓	0	631	22746	1519	1	1	0	22	0	1
This Work, $\gamma = 100000$	✓	419	0	426	676	3	1	0	1	0	1

Table 2. Concrete efficiency comparison of the most efficient UC-secure schemes for committing to messages of size $k = 256$, $\kappa = 128$ and $s = 40$ where the field is $\mathbb{F}_2$ and hence $\hat{s} = 40$. In the table γ represents the number of commitments the parties perform. These numbers include the cost of performing the initial OTs, both in terms of communication and computation.

To give a flavor of the actual numbers we compute Table 1 for specific parameters in Table 2. We fix the field to $\mathbb{F}_2$ and look at computational security $\kappa = 128$ and statistical security $s = 40$. As the schemes of [Lin11, BCPV13] rely on the hardness of the DDH assumption, a 256-bit EC group is assumed sufficient for 128-bit security [SRG$^+$14]. As already mentioned we look at message length $k = 256$ as this is well suited for these schemes.[3] The best code we could find for the schemes of [CDD$^+$15] in this setting has parameters [631,256,81] and is a shortened BCH code. For our scheme, the best code we have identified for the above parameters is a [419,256,40] expurgated BCH code [SS06]. Also, we recall the experiments performed in [CDD$^+$15] showing that exponentiations in a EC-DDH group of the above size require roughly 500 times more computation time compared to encoding using a BCH code for parameters of the above type.[4]

To give as meaningful comparisons as possible we also instantiate the initial OTs and include the cost of these in Table 2. As the homomorphic version of [CDD$^+$15] require 2-out-of-3 OTs in the setup phase, using techniques described in [LOP11, LP11], we have calculated that these require communicating 26 group elements and 44 exponentiations per invocation. The standard 1-out-of-2 OTs we instantiate with [PVW08] which require communicating 6 group elements and computing 11 exponentiations per invocation. In the table we do not take into consideration OT extension techniques [Bea96, IKNP03, Nie07, NNOB12, Lar15, ALSZ15, KOS15], as we do so few OTs that even the most efficient of these schemes *might* not necessarily improve the efficiency.

In the Table 2, we show results for different amounts of commitments. As can be expected our scheme improves as the number of committed values increases, but we note that at around 304 commitments, for the above message sizes and security parameters, our scheme outperforms all previous schemes in total communication and computation complexity, while at the same time offering additive homomorphism.

[3] We here assume a perfect efficient encoding of 256-bit values to group elements of a 256-bit EC group.

[4] They run the experiments with a shortened BCH code with parameters [796,256,121], which therefore suggests their observations are also valid for our choice of parameters.

References

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701, Sofia, Bulgaria, April 26–30, 2015. Springer, Berlin, Germany.

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 534–551, Banff, AB, Canada, June 25–28, 2013. Springer, Berlin, Germany.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 521–536, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[CDD$^+$15] Ignacio Cascudo, Ivan Damgård, Bernardo Machado David, Irene Giacomelli, Jesper Buus Nielsen, and Roberto Trifiletti. Additively homomorphic UC commitments with optimal amortized overhead. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 495–515, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Berlin, Germany.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[CvT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Berlin, Germany.

[DDGN14] Ivan Damgård, Bernardo Machado David, Irene Giacomelli, and Jesper Buus Nielsen. Compact VSS and efficient homomorphic UC commitments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 213–232, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Germany.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th ACM STOC*, pages 426–437, San Diego, California, USA, June 9–11, 2003. ACM Press.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 581–596, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[FJN$^+$13] Tore Kasper Frederiksen, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. MiniLEGO: Efficient secure two-party computation from general assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 537–556, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany.

[Fuj14] Eiichiro Fujisaki. All-but-many encryption - A new framework for fully-equipped UC commitments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 426–447, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Germany.

[GIKW14] Juan A. Garay, Yuval Ishai, Ranjit Kumaresan, and Hoeteck Wee. On the complexity of UC commitments. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 677–694, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 393–411, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Germany.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30, San Diego, California, USA, June 11–13, 2007. ACM Press.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Thomas Ristenpart, Rosario Gennaro, and Matthew Robshaw, editors, *CRYPTO 2015*, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Berlin, Germany.

[Lar15] Enrique Larraia. Extending oblivious transfer efficiently - or - how to get active security with constant cryptographic overhead. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 368–386, Florianópolis, Brazil, September 17–19, 2015. Springer, Berlin, Germany.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 446–466, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 259–276, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 329–346, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[Nao90] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 128–136, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[NFT09] Ryo Nishimaki, Eiichiro Fujisaki, and Keisuke Tanaka. Efficient non-interactive universally composable string-commitment schemes. In Josef Pieprzyk and Fangguo Zhang, editors, *ProvSec 2009*, volume 5848 of *LNCS*, pages 3–18, Guangzhou, China, November 11–13, 2009. Springer, Berlin, Germany.

[Nie07] Jesper Buus Nielsen. Extending oblivious transfers efficiently - how to get robustness almost for free. Cryptology ePrint Archive, Report 2007/215, 2007. `http://eprint.iacr.org/2007/215`.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[SRG$^+$14] Nigel P. Smart, Vincent Rijmen, Benedikt Gierlichs, Kenneth G. Paterson, Martijn Stam, Bogdan Warinschi, and Watson Gaven. Algorithms, key size and parameters report – 2014, 2014.

[SS06] Rudolf Schürer and Wolfgang Ch. Schmid. Mint: A database for optimal net parameters. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 457–469. Springer, 2006.

A Protocol Extension

As the scheme presented in Section 2 only implements commitments to random values we here describe an efficient extension to chosen message commitments. Our extension Π_{EHCOM} is phrased in the $\mathcal{F}_{\mathsf{HCOM}}$-hybrid model and it is presented in Fig. 6. The techniques presented therein are folklore and are known to work for any UC-secure commitment scheme, but we include them as a protocol extension for completeness. The **Chosen-Commit** step shows how one can turn a commitment of a random value into a commitment of a chosen value. This is done by simply using the committed random value as a one-time pad on the chosen value and sending this to P_r. The **Extended-Open** step describes how to open to linear combinations of either random commitments, chosen commitments or both. It works by using $\mathcal{F}_{\mathsf{HCOM}}$ to open to the random commitments and the commitments used to one-time pad the chosen commitments. Together with the previously sent one-time pad the receiver can then learn the designated linear combination.

Π_{EHCOM} describes a protocol between a sender P_s and a receiver P_r.

Chosen-Commit:

1. On input $(\texttt{chosen-commit},\texttt{sid},\texttt{cid},\boldsymbol{m})$, P_s picks an already committed to value $\boldsymbol{r}_j$ and computes $\widetilde{\boldsymbol{m}} = \boldsymbol{m} - \boldsymbol{r}_j$. It then sends $(\texttt{chosen},\texttt{sid},\texttt{cid},j,\widetilde{\boldsymbol{m}})$ to P_r. Else it ignores the message.
2. P_r stores $(\texttt{chosen},\texttt{sid},\texttt{cid},j,\widetilde{\boldsymbol{m}})$ and outputs $(\texttt{chosen-receipt},\texttt{sid},\texttt{cid})$.

Extended-Open:

1. On input $\left(\texttt{extended-open},\texttt{sid},\{(j,\alpha_j)\}_{j\in C_r},\{(l,\beta_l)\}_{l\in C_c}\right)$ with $\beta_l \in \mathbb{F}$ for $l \in C_c$ and $\alpha_j \in \mathbb{F}$, for $j \in C_r$, P_s verifies that and it has previously committed to a value $\boldsymbol{r}_j$ using $\mathcal{F}_{\mathsf{HCOM}}$ for $j \in C_r$. Else it ignores the message. For all $l \in C_c$ P_s verifies that it previously sent the message $\left(\texttt{chosen},\texttt{sid},l,\bar{j},\widetilde{\boldsymbol{m}}_l\right)$ to P_r. Let $\overline{\mathcal{J}}$ be the set of the corresponding indices $\bar{j}$, similarly let $\overline{\beta_{\bar{j}}} = \beta_l$ for the corresponding ID l. P_s then sends $\left(\texttt{open},\texttt{sid},\{(j,\alpha_j)\}_{j\in C_r} \cup \left\{\left(\bar{j},\overline{\beta_{\bar{j}}}\right)\right\}_{\bar{j}\in\overline{\mathcal{J}}}\right)$ to $\mathcal{F}_{\mathsf{HCOM}}$.
2. Upon receiving $\left(\texttt{open},\texttt{sid},\{(j,\alpha_j)\}_{j\in C_r} \cup \left\{\left(\bar{j},\overline{\beta_{\bar{j}}}\right)\right\}_{\bar{j}\in\overline{\mathcal{J}}},\boldsymbol{r}\right)$ from $\mathcal{F}_{\mathsf{HCOM}}$, P_r identifies the previously received messages $\left(\texttt{chosen},\texttt{sid},l,\bar{j},\widetilde{\boldsymbol{m}}_l\right)$ and outputs $\left(\texttt{extended-opened},\texttt{sid},\{(j,\alpha_j)\}_{j\in C_r},\{(l,\beta_l)\}_{l\in C_c},\boldsymbol{r}+\sum_{l\in C_c}\beta_l\cdot\widetilde{\boldsymbol{m}}_l\right)$.

Batch-Open:[a]

1. On input $(\texttt{batch-open},\texttt{sid},C_r,C_c)$. For all $j \in C_r$ P_s verifies that it has previously committed to a value $\boldsymbol{r}_j$ using $\mathcal{F}_{\mathsf{HCOM}}$. Else it ignores the message. For all $l \in C_c$ P_s verifies that it previously sent the message $\left(\texttt{chosen},\texttt{sid},l,\bar{j},\widetilde{\boldsymbol{m}}_l\right)$ to P_r. P_s then sends $\left(\texttt{batch-open},\texttt{sid},\{(j,\boldsymbol{r}_j)\}_{j\in C_r},\{(l,\boldsymbol{m}_l)\}_{l\in C_c}\right)$ to P_r, where $\boldsymbol{r}_j$ and $\boldsymbol{m}_l$ are random and chosen messages, respectively, previously committed to.
2. Let $t_r = |C_r|$ and $t_c = |C_c|$. P_r then samples uniformly random values $x_1,\ldots,x_{t_r},y_1,\ldots,y_{t_c} \in \mathbb{F}$ and sends these to P_s.
3. P_s and P_r run **Extended-Open** with input
$$\left(\texttt{extended-open},\texttt{sid},\{(j_u,x_u)\}_{u\in[t_r]},\{(l_v,y_v)\}_{v\in[t_c]}\right)$$
where j_u and l_v are the u'th and v'th element of C_r and C_c respectively, under an arbitrary ordering.
4. P_r lets $\left(\texttt{extended-opened},\texttt{sid},\{(j_u,x_u)\}_{u\in[t_r]},\{(l_v,y_v)\}_{v\in[t_c]},\boldsymbol{n}\right)$ be the output of running **Extended-Open**. P_r now verifies that
$$\boldsymbol{n} = \sum_{u\in[t_r]} x_u \cdot \boldsymbol{r}_{j_u} + \sum_{v\in[t_c]} y_v \cdot \boldsymbol{m}_{l_v} \ .$$
If true then P_r outputs $\left(\texttt{batch-opened},\texttt{sid},\{(j,\boldsymbol{r}_j)\}_{j\in C_r} \cup \{(l,\boldsymbol{m}_l)\}_{l\in C_c}\right)$. Else it aborts and halts.

[a] The following steps assumes that $|\mathbb{F}|^{-1} \leq 2^{-s}$. If this is not the case, the test will be repeated a number $\hat{s}$ times such that $|\mathbb{F}|^{-\hat{s}} \leq 2^{-s}$. This requires generating a factor $\hat{s}$ additional challenges.

Fig. 6. Protocol Π_{EHCOM} in the $\mathcal{F}_{\mathsf{HCOM}}$-hybrid model.

Finally we present a **Batch-Open** step that achieves very close to optimal amortized communication complexity for opening to a set of messages. The technique is similar to the consistency check of Π_{HCOM}. When required to open to a set of messages, the sender P_s will start by sending the messages directly to the receiver P_r. Next, the receiver challenges the sender to open to a random linear combination of all the received messages. When receiving the opening from $\mathcal{F}_{\mathsf{HCOM}}$, P_r verifies that it is consistent with the previously received messages and if this is the case it accepts these. For the exact same reasons as covered in the proof of Theorem 1 it follows that this approach of opening values is secure. For clarity and ease of presentation the description of batch-opening does not take into account opening to linear combinations of random and chosen commitments. However the procedure can easily be extended to this setting using the same approach as in **Extended-Open**.

In terms of efficiency, to open N commitments with message-size l, the sender needs to send lN field elements along with the verification overhead $\hat{s}\hat{O}+\kappa$ where $\hat{O}$ is the cost of opening to a commitment using $\mathcal{F}_{\mathsf{HCOM}}$. Therefore if the functionality is instantiated with the scheme Π_{HCOM}, the total communication for batch-opening is $\hat{s}(k+n)f+\kappa+kNf$ bits where k is the length of the message, n is the length of the code used, f is the size of a field element and $\hat{s}$ is the number of consistency checks needed.

We now elaborate on the applicability of batch-opening for committing to large messages as mentioned in Section 2.3. Recall that there we split the large message m of size M into N blocks of size l and the idea is to instantiate Π_{HCOM} with a $[n_s,l,s]$ code and commit to m in blocks of size l. This requires n_s initial OTs to setup and requires sending $(2\hat{s}n_s)f+\kappa+lNf$ bits to commit to all blocks. For a fixed s this has rate close to 1 for large enough l. In the opening phase we can then use the above batch-opening technique to open to all the blocks of the original message, and thus achieve a rate of ${}^{Mf}/_{\hat{s}(l+n_s)f+\kappa+lNf}\approx 1$ in the opening phase as well.

In [GIKW14] the authors present an example of committing to strings of length 2^{30} with statistical security $s=30$ achieving rate $1.046^{-1}\approx 0.95$ in both the commit and open phase. To achieve these number the field size is required to be very large as well. The authors propose techniques to reduce the field size, however at the cost of reducing the rate. We will instantiate the approach described above using a binary BCH code over the field $\mathbb{F}_2$ and recall that these have parameters $[n-1,n-\lceil\frac{d-1}{2}\rceil\log(n+1),\geq d]$. Using a block length of 2^{13} and $s=30$ therefore gives us a code with parameters $[8191,7996,30]$. Thus we split the message into $134{,}285=\lceil{}^{2^{30}}/_{7996}\rceil$ blocks. In the commitment phase we therefore achieve rate ${}^{2^{30}}/_{2\cdot 30\cdot 8191+128+8191\cdot 134{,}285}\approx 0.976$. Using the batch-opening technique the rate in the opening phase is even higher than in the commit phase, as this does not require any “blinding” values. In the above calculations we do not take into account the 8191 initial OTs required to setup our scheme. However using the OT-extension techniques of [KOS15], each OT for κ-bit strings can be run using only κ initial “seed” OTs and each extended OT then requires only κ bits of communication. Instantiating the seed OTs with the protocol of [PVW08] for $\kappa=128$ results in $6\cdot 256\cdot 128+8191\cdot 128=1{,}245{,}056$ extra bits of communication which lowers the rate to 0.974.

Finally, based on local experiments with BCH codes with the above parameters, we observe that the running time of an encoding operation using the above larger parameters is roughly 2.5 times slower than an encoding using a BCH code with parameters [796,256,121]. This suggests that the above approach remains practical for implementations as well.