1. Abstract: needs rewriting
2. Motivation:

* Citation needed for german elections?
* Good, was proofread
* But: Maybe check back regarding comparison: Not a lot of comparison done to other SMPC schemes

1. Cryptographic Basics
   1. TODO: Check spelling, writing, is everything necessary?
2. SMPC BASICS & HISTORY
   1. Poker/Millionaire problems: add source
   2. Terminology
      1. ~~OT: Remove?~~
      2. ~~Adversary model: ADD~~
   3. Otherwise: good, was proofread/\*
   4. Add Stuff regarding SPDZ, BDOZ, SPDZ 2.0
   5. Shamir secret sharing: Add link to SPDZ
   6. ~~TALK about paradigm: share, compute, reveal~~
3. SCALE-MAMBA
   1. Secret sharing

Online phase

* ~~Remove beaver~~
* Otherwise decent

How is tag calculated with shared mac?

Add runtime to offline phase

Offline phase: Go more into detail about the mac key, how is it distributed?

~~Beaver triples: Maybe move creation to later part?~~

~~Maybe put online before offline phase?~~

Triple sacrificing was done before in online or offline phase??

Fix Offline phase ZKPOK: Overdirve, TopGear etc.

Give more examples where it is used, how the runtime compares

Give a short comparison to other SMPC schemes, when SCALE-MAMBA should be applied

Add all the necessary references

Add: this paper didn’t give a complete overview, but tries to present a selective introduction to provide a general understand of SCALE-MAMBA, its functionalities, why and how it works, and when it can be used and utilized

A lot of these protocols use secret sharing based on so called Semi-Homomorphic Encryption(SHE). Although it is possible to make a fully homomorphic secret sharing, which implies that it addition and multiplication keep the the secret sharing intact and the result can be directly recovered, these schemes are highly computational complex. SHE based protocols

\section{Cryptographic Basics}\label{sec:basics}

Before delving deeper into SMPC, this section will shortly recap on cryptography basics that might be unknown to the reader. These basics are not solely used by SMPC, but nevertheless, play a huge role in a lot of protocols.

\subsection{Message Authentification Codes}

Message Authentification Codes (MACs) are used to guarantee the integrity and authenticity of a message. Basically, a MAC is a key that is used to generate a tag for a message. This tag is used to verify the identity of the sender and that the content of the message was not changed.\\

For using MACs we need a tupel of two algorithms \((\text{MAC},\text{VER})\):\\

\begin{itemize}

\item The MAC algorithm \(\text{MAC}(K,m)\rightarrow t\) takes a key \(K\) and a message \(m\) and outputs a tag \(t\)\\

\item The verficiation algorithm \(\text{VER}(K,m,t)\rightarrow 0/1\) takes a key \(K\), a message \(m\) and a tag \(t\) and outputs accept(\(1\)) or reject(\(0\))\\

\end{itemize}

The correctness can be easily seen: \(\forall K: \text{VER}(K,m,\text{MAC}(K,m))=1\)\\

In SCALE-MAMBA really simple MACs are used. We assume that we have a MAC key \( \alpha \). Then the MAC algorithm used is following:\\

$$\text{MAC}(\alpha,m)\rightarrow \alpha \* m$$

$$\text{VER}(\alpha,m,t)\rightarrow (1 \Leftrightarrow t = \alpha\*m) $$

This algorithm has the disadvantage of high storage complexity, but therefore it doesn't take a long time to calculate and it is symmetric so the same key can be used for creating the tag and verifying it. The symmetry is used for an advantage later on in the protocol.\\

\subsection{Homomorphic Encryption}

Encryption as a basic concept is simple. Similiar to MACs, a tupel (E,D) of algorithms is needed.\\

\begin{itemize}

\item The enncryption algorithm \(\text{E}(K,m)\rightarrow c\) takes a key \(K\) and a message \(m\) and outputs a cyphertext \(t\) that is the encryption of the message \(m\)\\

\item The decryption algorithm \(\text{D}(K,c)\rightarrow m\) takes a key \(K\) and a cyphertext \(m\) and outputs the decrypted message \(m\), assuming c was not modified\\

\end{itemize}

Alternatively, in a public key setting, the key for encryption is public and there is a different key for decryption that is privat.\\

The problem is, that modifying the cypertext in any way makes the decryption useless. In some cases it is desirable do do computations on cypertexts, because then it is possible to make calculations without revealing the unencrypted data. That is why Homomorphic Encryption was developed. Let's assume that we have a ring \((R,+,\*)\) as our message space. Fully Homomorphic Encryption (FHE) is preserving the ring structure. Therefore, it is possible to do additions and multiplications of cypertexts and when decrypting the result it is the same value as when doing the same additions and multiplications on the real values.\\

Althought FHE has amazing properties for a lot of usecases, in praxis it is way too slow. Because of all the overhead generated by the encryption scheme to guarantee the preservation of the ring structure, the current algorithms are not feasible for most applications.

\subsection{Somewhat Homomorphic Encryption}

That is where Somewhat Homomorphic Encryption (SHE) becomes relevant. SHE only supports one operation, and in most cases that is addition. This takes away a lot of possibilities but, therefore, the performance gain is immense. So immense, that SHE is used a lot in practice, for example in SCALE-MAMBA.\\

The SHE scheme used in SCALE-MAMBA, is based on the so called Brakerski-Gentry-Vaikuntanathan cryptosystem(BGV).\\