When mutually distrustful parties wish to compute some joint function of their private inputs, they require a certain number of security properties to hold for that computation:

- Privacy: Nothing is learnt from the protocol besides the output;
- Correctness: The output is distributed according to the prescribed functionality;
- Independence: One party cannot make its inputs depend on the other parties’ inputs;
- Delivery: An adversary cannot prevent the honest parties from successfully computing the functionality.
- Fairness: If one party receives output then so do all.

Any multi-party computation can be securely computed as long as there is a honest majority. If there is no such majority, and in particular in the two-party case, it is impossible to achieve both fairness and guaranteed output delivery.

**Objectives**

We describe a new contract signing protocol that achieves a new notion of fairness and abuse-freeness. This protocol is based on the well-known Schnorr signature protocol. The new contract signing protocol is provably secure in the random oracle model under the hardness assumption of solving the discrete logarithm problem. This construction can be adapted to other IBL schemes.

**Schnorr Signatures**

Schnorr digital signatures are an offspring of ElGamal signatures. To generate signature keys select large primes \( p, q \) such that \( p - 1 \mod q = 0 \), as well as an element \( g \in \mathbb{G} \) of order \( q \). A hash function \( H : \{0,1\}^* \rightarrow \mathbb{G} \). The output is a set of public parameters \( pp = (g, p, q, H) \). Select at random \( x \in \mathbb{Z}_q \) and compute \( y = g^x \). The output is the couple \((sk, pk)\) where \( sk = x \) is kept private, and \( pk = y \) is made public. To sign a message \( m \) select a random \( r \in \mathbb{Z}_q \) and compute:

\[ r = g^k, \quad e = H(m || r), \quad s = k - e \mod q \]

and outputs \((r, s)\) as the signature of \( m \).

To verify the signature compute \( e = H(m || r) \) and check that \( g^{sy} = g^{ry} \).

**Schnorr Co-Signatures**

Schnorr’s signatures can be generalized to two signers. This produces co-signatures, i.e. a signature formed by joining forces between two signers.

**How Does It Work?**

We now address a subtle weakness in the protocol described in the previous section, which is not captured by the fairness property per se and that we refer to as the existence of “proofs of involvement.” Such proofs are not valid co-signatures, and would not normally be accepted by verifiers, but they nevertheless are valid evidence establishing that one party committed to a message. In a legally fair context, it may happen that such evidence is enough for one party to win a trial against the other — who lacks both the co-signature, and a proof of involvement.

To enforce fairness on the co-signature protocol, we ask that the equivalent of a keystone is transmitted first, so that in case of dispute, the aggrieved party has a legal recourse. First we define the notion of an authorized signatory credential:

**Definition** (Authorized signatory credential) The data field 

\[ \Gamma_{\text{authorized}} = (\text{Alice, Bob, } k_A, k_B, \sigma_A(g^{k_A}(\text{Alice})B_{\text{Bob}})) \]

is called an authorized signatory credential given by Alice to Bob, where \( \sigma_A(g^{k_A}) \) is some publicly known auxiliary signature algorithm using Alice’s private key \( k_A \) as a signing key. Any party who gets \( \Gamma_{\text{authorized}} \) can check its validity, and releases \( \Gamma_{\text{authorized}} \) to be convention functionally equivalent to Alice giving her private key \( k_A \) to Bob. A valid signature by Bob on a message \( m \) validated with a valid \( \Gamma_{\text{authorized}} \) is legally defined as encompassing the meaning (\( \Delta(m) \)) of Alice’s signature on \( m \):

\[ \Gamma_{\text{authorized}} \\\text{signature by Bob on m} \equiv \text{signature by Alice on m} \]

Second, the co-signature protocol is modified by requesting that Alice provide a valid \( \Gamma_{\text{authorized}} \) to Bob. Bob stores this in a local non-volatile memory \( C \) along with \( g^{k_B} \). For all practical purposes, \( C \) can be simply regarded as Bob’s hard disk. Together, \( f\) and \( g^{k_B} \) act as a keystone enabling Bob (or a verifier, e.g. a court of law) to reconstruct \( \Gamma_{\text{authorized}} \) if Alice exhibits a (translucent) signature binding Bob along with his co-signing public key.

Therefore, should Alice try to exhibit a signature of Bob alone on a message they both agreed upon (which is known as a fault), the court would be able to identify Alice as the fraudster. The resulting protocol is:

**Legal Fairness**

The main idea builds on the following observation: Every signature exchange protocol is plagued by the possibility that the last step of the protocol is not performed. Indeed, it is in the interest of a malicious party to get the other party’s signature without revealing its own. As a result, the best one can hope for is that a trusted third party can eventually notice the stop and drop the protocol early, which at least avoids the adversary getting a signature from the party at the end of the protocol, and vice versa. Instead, we construct a joint signature, or co-signature, of both Alice and Bob. By design, there are no signatures to steal — and stopping the protocol early does not give the stopper a decisive advantage.

**Introduction**

In two-party computation, achieving both fairness and guaranteed output delivery is well known to be impossible. In this paper we describe and analyze a new contract signing paradigm called legal fairness. This paradigm is very close to fairness and is realizable. We give a concrete legal fairness protocol based on Schnorr signatures. The new protocol is provably secure in the random oracle model under the DLP assumption.